A

DEPARTMENT OF THE NAVY

OFFICE OF THE GENERAL COUNSEL

COUNSEL FOR THE
NAVAL RESEARCH LABORATORY
WASHINGTON, D.C. 20375-5325

PATENTS
5870
3008.2
31 Dec 97

Navy Case No. 78,824

Commissioner of Patents and Trademarks
Washington, D. C. 20231

Applicant(s): James H. Adams, Jr., Paul Boberg, Buddy Brownstein, William Dietrich, Erwin Flueckiger, Edward Petersen, Margaret Shea, Don Smart, and Edward Smith

For: **METHOD AND APPARATUS FOR MODELING COSMIC RAY EFFECTS ON MICROELECTRONICS**

Sir:

Transmitted herewith are the papers above-identified constituting a Patent Application filed by the Department of the Navy on behalf of the above-names Applicant(s).

The total filing fee has been computed in accordance with the following formula:

Basic Fee ---------------------------------$790.00

Additional Fees:
1. Total number of claims in excess of 20 times $22.00---------------$_____
2. Number of independent claims minus 3 times $82.00--------------$246.00

Total Filing Fee---------------------------$1036.00

Kindly charge the aforementioned total filing fees and any additional fees due, or credit overpayment of fees, to Deposit Account No. 04-0814.

Respectfully,

Thomas E. McDonnell

THOMAS E. McDONNELL
Reg. #26950

Encl:

DEPARTMENT OF THE NAVY

OFFICE OF THE GENERAL COUNSEL

COUNSEL FOR THE
NAVAL RESEARCH LABORATORY
WASHINGTON, D.C. 20375-5325

5870
3008.2
31 DEC 97

Navy Case No. 78,824

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application: John H. Adams, et al.

For: **METHOD AND APPARATUS FOR MODELING COSMIC RAY EFFECTS ON MICROELECTRONICS**

NOTICE OF FILING WITHOUT OATH OR DECLARATION BY APPLICANT

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Sir:

Please take notice that this application is being filed pursuant to 37 C.F.R. 153(d) without the oath or declaration of the Applicant. This is being done to secure an early filing date.

Upon the Notice from the Patent Office required by 1.53(d), Applicant will file the required oath or declaration, and pay the surcharge as set forth in 37 C.F.R. 1.16(e).

Respectfully submitted,

Thomas E. McDonnell

THOMAS E. McDONNELL
Reg. No. 26950
Date:

DEPARTMENT OF THE NAVY

OFFICE OF THE GENERAL COUNSEL

Navy Case No. 78,824

COUNSEL FOR THE
NAVAL RESEARCH LABORATORY
WASHINGTON, D.C. 20375-5325

APPLICATION FOR LETTERS PATENT

TO ALL WHOM IT MAY CONCERN:

BE IT KNOWN THAT **James H. Adams, Jr., Paul Boberg, Buddy Browstein, William Dietrich, Erwin Flueckiger, Edward Petersen, Margaret Shea, Don Smart, Edward Smith and Allen Tylka** are citizens of the United States of America, and residents of Alexandria, Va, College Park, Md, Oxen Hill, Md, Chicago, IL, Bern Switzerland, Fairfax, Va, Bedford, Ma, Bedford, Ma, Mounteagle, TN, Silver Spring, Md, have invented certain new and useful improvements in **METHOD AND APPARATUS FOR MODELING COSMIC RAY EFFECTS ON MICROELECTRONICS** of which the following is a specification:

Prepared by:
John J. Karasek
Reg. No. 36,182
Tele. No. (202) 767-3427

METHOD AND APPARATUS FOR MODELING COSMIC RAY EFFECTS ON MICROELECTRONICS

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates generally to a method and apparatus for modeling the effects of cosmic rays on microelectronics in earth orbit, and more particularly to an improved method and apparatus for modeling these effects, these improvements reflecting a simpler, easier way for the user to operate the model over the internet, and more accurate modeling of these effects.

Description of the Related Art

For electronic components onboard satellites in earth orbit, exposure to cosmic rays represents a serious risk, due to the capacity of cosmic rays to induce single event effects (SEE) in these components. *See generally* Sherra E. Kerns, *Transient-Ionization and Single-Event Phenomena, in* IONIZING RADIATION EFFECTS IN MOS DEVICES AND CIRCUITS 485-91 (John Wiley & Sons, Inc., T.P. Ma et al. eds., 1989). In brief, SEEs occur when an energetic particle changes a particular device in an integrated circuit, thereby causing an error. To date, the only effective methods for hardening these circuits against SEEs have been shielding, redundancy, and error detection and correction (EDAC).

Unfortunately, these measures are not always effective, and the failure of a single component may lead to the total loss of a multi-million dollar satellite. With the potential loss associated with this risk so high it is desired to have a way to accurately predict the magnitude of this risk, as well as a way of predicting how successful ameliorative efforts are likely to be.

CREME (short for Cosmic Ray Effects on Microelectronics) was a software package developed by the Naval Research Laboratory in 1981 for modeling how a given electronic chip on a given satellite with a given orbit and amount of shielding would hold up against cosmic ray bombardment. *See* Cosmic Ray Effects on Microelectronics, Part I: The Near-Earth Particle Environment, Adams et al., NRL Memorandum Report 4506; Cosmic Ray Effects on Microelectronics Part II: The Geomagnetic Cutoff Effects, Adams et al., NRL Memorandum Report 5099; Cosmic Ray Effects on Microelectronics, Part IV, Adams, NRL Memorandum Report 5901, each incorporated herein by reference, in their entireties, for all purposes. CREME had several shortcomings. It has been discovered that many of the predictions of CREME were inaccurate. Particular shortcomings of the original CREME software included its inaccurate modeling of the transmission of cosmic rays through earth's magnetosphere, and inaccurate modeling of the flux of heavy ions associated with solar flares.

Moreover, the implementation of this program was less than optimal, having a difficult user interface, and requiring each user to install, maintain, and run the software.

NOTE ON NOMENCLATURE USED HEREIN

Some portions of the following disclosure describe method elements such as determining, selecting, dividing, processing, computing, calculating, numerically integrating, applying a function, displaying, and the like, and structure elements such as tables, memories, internet connections, and the like. These descriptions are the means used by persons of ordinary skill in the art of data processing to most effectively convey the substance of their work to other skilled artisans. Such methods and structures are intended to describe methods and structures for carrying out a set of steps on at least one programmed digital computer to reach a desired result. Thus, each of these steps requires a physical manipulation of concrete quantities, generally in the form of electrical, optical, and magnetic signals capable of being stored, retrieved, combined, and otherwise manipulated in such a programmed digital computer.

Accordingly, unless indicated otherwise, skilled artisans will recognize that as used herein, terms such as determining, selecting, dividing, processing, computing, calculating, numerically integrating, applying a function, displaying, and the like, refer to the operations of a programmed digital computer system, or similar electronic computing device, that manipulates and transforms data represented as physical quantities within the computer.

SUMMARY OF THE INVENTION

Accordingly, it is an object of this invention to improve the modeling of the effects of cosmic rays on microelectronics.

It is a further object of this invention to improve the modeling of solar heavy ion flux.

It is a further object of this invention to improve the modeling of geomagnetic transmission.

It is a further object of this invention to improve the user interface, by implementing an internet-based interface, allowing users to access a world wide web site connected to a server where the software is installed and maintained.

These and additional objects of the invention are accomplished by the structures and processes hereinafter described.

An aspect of the present invention is a method and apparatus for computing a geomagnetic transmission function. This apparatus includes a programmed digital computer running modeling software for modeling the transmission of cosmic ray particles through the magnetosphere. The software includes a model representing a solution to the Lorentz equation in a magnetic field given by:

$$\mathbf{B} = \mathbf{B}_{IGRF}(\mathbf{r},t') + \mathbf{B}_{TSYG}(Kp,\mathbf{r},t')$$

where **B** is the earth's magnetic field, where $\mathbf{B}_{IGRF}(\mathbf{r},t')$ is the International Geomagnetic Reference Field model of earth's magnetic field, a standard internationally recognized representation of earth's magnetic field, and where $\mathbf{B}_{TSYG}(Kp,\mathbf{r},t')$ is the model of earth's magnetic field published by Tsyganenko in 1989, as modified by the inventors. $\mathbf{B}_{IGRF}(\mathbf{r},t')$ and $\mathbf{B}_{TSYG}(Kp,\mathbf{r},t')$ are discussed in further detail *infra*.

Another aspect of the present invention is a method and apparatus for computing a flux of particles at the outer surface of a satellite comprising, *inter alia*, an improved method and apparatus for computing a flux of solar heavy ions. This apparatus includes a programmed digital computer

running modeling software for modeling the flux of cosmic ray particles through the outer surface of a satellite.

Another aspect of the invention is a method and apparatus comprising a programmed digital computer running modeling software for modeling the effect of cosmic rays on microelectronics, where this software embodies at least one of the two foregoing aspects of the invention.

Another aspect of the invention is a preferred embodiment of a method and apparatus comprising a programmed digital computer running modeling software for modeling the effect of cosmic rays on microelectronics, where this software embodies at least one of the two foregoing aspects of the invention, where this preferred embodiment is connected to a network, typically the internet, to permit remote users to use the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete appreciation of the invention will be obtained readily by reference to the following Description of the Preferred Embodiments and the accompanying drawings in which like numerals in different figures represent the same structures or elements, wherein:

FIG. 1 is a partial flowchart for a preferred embodiment of the invention.

FIG. 2 is an elevation of the earth, showing the path of a cosmic ray particle in the magnetosphere.

FIG. 3 is a flowchart for a method for calculating a geomagnetic transmission function for cosmic rays according to the invention.

FIG. 4 is a plot of a geomagnetic transmission function for a low inclination space shuttle orbit under quiet magnetospheric conditions.

FIG. 5 is a plot of a geomagnetic transmission function for a lo inclination space shuttle orbit under stormy magnetospheric conditions.

FIG. 6 is a plot of a geomagnetic transmission function for a space station orbit under quiet magnetospheric conditions.

FIG. 7 is a plot of a geomagnetic transmission function for a space station orbit under stormy magnetospheric conditions.

FIGS. 8 and 9 together are a flowchart for a method for modeling solar heavy ion flux according to the present invention.

FIG. 10 shows a main menu for a user to operate the preferred CREME96 software through the internet.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The following are each incorporated herein by reference, in their entireties, for all purposes:

(A) "CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code", Transactions on Nuclear Science **44**(6) 2150-60, Tylka et al. (1997);

(B) "A Magnetosphereic Magnetic Field Model with a Warped Tail Current Sheet", Planet. Space Sci. **37**(1) 5-20, Tsyganenko (1989).

Overview

In a preferred embodiment, the invention comprises several computer software modules running on a powerful server computer. Although this computer may be operated in a standalone mode, it is preferred to have this server connected to a network (typically the internet), so that remote users can operate the software, and store their resulting data either locally on the server or remotely on their own or another machine.

Overall, the software will include one or both of a geomagnetic transmission modeling module and a particle flux modeling module. Optionally, the software includes both of these modules, and also optionally the software includes one or more of several optional modules, including a shielding transport modeling module, a linear energy transfer modeling module, a proton-induced single event upset (SEU, a particular type of SEE, but unless noted otherwise used interchangeably herein with SEE) rate modeling module, and a direct ionization induced single event upset rate modeling module.

Examples of each of these modules have been produced, and will be discussed in greater detail below. The exemplary module for modeling geomagnetic transmission is referred to herein as GTRN. The exemplary module for modeling particle flux is referred to herein as FLUX. The exemplary module for modeling transport of particles through solid shielding is referred to herein as TRANS. The exemplary module for modeling linear energy transfer to a target structure (typically an integrated circuit) is referred to herein as LETSPEC. The exemplary module for modeling proton-induced SEU event upset rates in specified devices is referred to herein as PUP. The exemplary module for modeling SEU rates induced by direct ionization is referred to herein as HUP.

Exemplary FORTRAN code for these exemplary modules appears *infra*. Together, the software for these exemplary modules is referred to herein as CREME96.

Referring to FIG. 1, one sees that CREME96 produces two types of outputs: proton-induced SEU rates, and direct ionization induced SEU rates. Each of these will represent an upset rate for a given electronic device in a satellite with a given shielding, under given environmental conditions, for a satellite in a given orbit or orbit segment.

Referring to **100**, in the case where the specified satellite orbit is at or above a given altitude, nominally geosynchronous, the software will evaluate the fluxes of SEU-inducing particles at the outer surface of the satellite, from a number of sources including galactic cosmic rays (GCR), anomalous cosmic rays (ACR), and solar heavy ions. The environment at such satellite orbits is referred to herein as the "non-trapped energetic particle environment", since it is essentially equivalent to the environment at other locations that are at about 1 AU from the sun, but outside of the volume where trapped or quasi-trapped particles would be found in significant numbers.

Referring to **200**, in the case where the specified satellite orbit is below a given altitude, nominally geosynchronous, before the flux at the outer surface of the satellite is evaluated, the shielding effect of the earth's magnetosphere is evaluated. As persons of ordinary skill in the art will recognize, the earth's magnetic field will shield out a significant fraction of charged particles. Thus, it is necessary to first subtract from the flux in the non-trapped energetic particle environment that fraction of the flux that will be shielded by the magnetosphere.

Referring to **300**, the external hull of a satellite (typically made from thin aluminum panels) will shield the electronics of the satellite from a fraction of SEU-inducing particles. Thus, determining an SEU rate for microelectronics on a given satellite will require taking this shielding into account by correcting the flux for the energy lost in said shielding and for the fragmentation of the incident particles within said shielding, and subtracting from the flux of SEU-inducing particles the subset of SEU-inducing particles with energies that are not sufficient to penetrate through the hull of the satellite. This module of the software performs that function. Although, as indicated by the dashed lines in the flowchart, this module may be bypassed, it is highly preferred to *not* bypass this module.

Referring to **400**, although an approximation of the satellite hull shielding may be obtained by assuming that the shielding is uniformly distributed about the electronics of interest, more realistic results may be obtained by specifying the particular distribution of shielding about the electronics for a particular satellite. This module of the software performs that function.

Referring to **500**, after the software measures the transmission of SEU-inducing particles, the software will measure the effect of SEU-inducing protons on the electronics of interest.

Referring to **600**, after the software measures the transmission of SEU-inducing particles, the software will calculate the linear energy transfer to the electronics of interest. This module of the software takes an input file in the form of a particle flux vs. kinetic energy table for each elemental species, and generates an output file in the form of a function specifying the combined flux of all

these species as a function of the rate of energy deposition. This is an intermediate result for the calculation of SEU rates attributable to direct ionization.

Referring to **700**, this program module takes the output linear energy transfer spectrum from **600**, combines this with parameters for the particular electronic device, calculates a SEU rate for the device attributable to direct ionization. The parameters are typically based on ground tests of the sensitivity to a particular chip to SEU-inducing radiation.

Having given an overview of this preferred embodiment of the invention, a more detailed description of the particular software modules making up this preferred embodiment is given.

Geomagnetic Transmission Function

Although it has been known that the magnetosphere shields cosmic rays, quantifying this effect to a degree of accuracy that could lead to accurate models of the effects of cosmic rays on electronics within the magnetosphere has proven elusive. One difficulty has been the lack of accurate models for the magnetosphere. Another difficulty has been that even if an accurate model of the magnetosphere were available, modeling the shielding effect of the magnetosphere requires an inordinate amount of computer processing time.

Referring to FIG. 2, this figure shows a representative path of a cosmic ray through the earth's magnetosphere (not shown). This path is highly irregular, including numerous changes of direction, and covering widely-spaced areas over earth's surface. Determining whether a particle

with such a convoluted path will intersect a body as small as a typical satellite is a considerable challenge.

$\mathbf{B}_{IGRF}$ is the International Geomagnetic Reference Field promulgated by the International Association of Geomagnetism and Aeronomy. It is the internationally accepted standard expression for the earth's internal magnetic field. It has been discovered that the $\mathbf{B}_{IGRF}$ is insufficient, and that the most well-known model of the magnetospheric **B** fields, the model provided by Tsyganenko et al., "A Magnetospheric Magnetic Field Model with a Warped Tail Current Sheet", *Planet. Space Sci.*, **37**(1) 5-20 (1989), is likewise insufficient. Corrections to this model are included in the present invention.

It has been discovered that within the earth's magnetosphere, the total magnetic field can be modeled by;

$$\mathbf{B} = \mathbf{B}_{IGRF}(\mathbf{r},t') + \mathbf{B}_{TSYG}(Kp,\mathbf{r},t')$$

where $\mathbf{B}_{IGRF}$ is the International Geomagnetic Reference Field promulgated by the International Association of Geomagnetism and Aeronomy, and $\mathbf{B}_{TSYG}$ is the modified Tsyganenko field given by the sum of $\mathbf{B}_{Xmg}{}^{(T)} + \mathbf{B}_{Ymg}{}^{(T)} + \mathbf{B}_{Zmg}{}^{(T)} + \mathbf{B}_{Xmg}{}^{(RC)} + \mathbf{B}_{Ymg}{}^{(RC)} + \mathbf{B}_{Zmg}{}^{(RC)} + \mathbf{B}_{Xsm}{}^{(C)} + \mathbf{B}_{Ysm}{}^{(C)} + \mathbf{B}_{Zsm}{}^{(C)} + \mathbf{B}_{Xsm}{}^{(M)} + \mathbf{B}_{Ysm}{}^{(M)} + \mathbf{B}_{Zsm}{}^{(M)}$, where coordinates in the solar magnetospheric system are denoted sm, and where coordinates in the solar magnetic coordinate system are denoted mg; and where

$$\mathbf{B}_{Xsm}{}^{(T)} = Q_t\, x_{sm}\, z_r;$$

$$\mathbf{B}_{Ysm}{}^{(T)} = Q_t\, y_{sm}\, z_r;$$

$$\mathbf{B}_{Zsm}(T) = \frac{W(x,y)}{S_T}\left(C_1 + C_2\frac{a_T + \xi_T}{S_T^2}\right) + \frac{x\frac{\partial W}{\partial x} + y\frac{\partial W}{\partial y}}{S_T + a_T + \xi_T} \times$$

$$(C_1 + C_2/S_T) + \mathbf{B}_X(T)\frac{\partial z_s}{\partial x} + \mathbf{B}_Y(T)\frac{\partial z_s}{\partial y} - Q_T D_T\left(x\frac{\partial D_T}{\partial x} + y\frac{\partial D_T}{\partial y}\right)$$

where $W(x,y) = 0.5\left(1 - \frac{x - x_0}{\left[(x-x_0)^2 + D_x^2\right]^{1/2}}\right)$

where $Q_T = \frac{W(x,y)}{\xi_T S_T}\left[\frac{C_1}{S_T + a_T + x_T} + \frac{C_2}{S_T^2}\right]$;

where $z_T = z - z_s(x,y,\psi)$,

$$z_s(x,y,\psi) = 0.5\tan\psi\left(x+R_c-\sqrt{(x+R_c)^2+16}\right) - G\sin\psi\cdot y^4\left(y^4+L_y^4\right)^{-1},$$

where
$$S_{T,RC} = \sqrt{\rho^2+\left(a_{T,RC}+\xi_{T,RC}\right)^2},$$
$$\xi_{T,RC} = \sqrt{z_r^2+D_{T,RC}^2},$$
$$D_T = D_0+\delta y^2+\gamma_T h_T(x)+\gamma_1 h_1(x),$$

where $C_1 = -98.72$ when $Kp = 0, 0^-$, $C_1 = -35.64$ when $Kp = 1^-,1,1^+$, $C_1 = -77.45$ when $Kp = 2^-,2,2^+$, $C_1 = -70.12$ when $Kp = 3^-,3,3^+, 0, 0^+$, $C_1 = -162.5$ when $Kp = 4^-,4,4^+$, $C_1 = -128.4$ when $Kp \geq 5^-$,

where $C_2 = -10014$ when $Kp = 0, 0^-$, $C_2 = -12800$ when $Kp = 1^-,1,1^+$, $C_2 = -14588$ when $Kp = 2^-,2,2^+$, $C_2 = -16125$ when $Kp = 3^-,3,3^+, 0, 0^+$, $C_2 = -15806$ when $Kp = 4^-,4,4^+$, $C_2 = -16184$ when $Kp \geq 5^-$,

where $a_T = 13.55$ when $Kp = 0, 0^+$, $a_T = 13.81$ when $Kp = 1^-,1,1^+$, $a_T = 15.08$ when $Kp = 2^-,2,2^+$, $a_T = 15.63$ when $Kp = 3^-,3,3^+, 0, 0^+$, $a_T = 16.11$ when $Kp = 4^-,4,4^+$, $a_T = 15.85$ when $Kp \geq 5^-$,

where $D_0 = 2.08$ when $Kp = 0, 0^+$, $D_0 = 1.664$ when $Kp = 1^-,1,1^+$, $D_0 = 1.541$ when $Kp = 2^-,2,2^+$, $D_0 = 0.9351$ when $Kp = 3^-,3,3^+, 0, 0^+$, $D_0 = 0.7677$ when $Kp = 4^-,4,4^+$, $D_0 = 0.3325$ when $Kp \geq 5^-$,

where $R_c = 8\ R_E$ and $L_y = 10\ R_E$,

$\mathbf{B}_X^{(RC)} = Q_{RC} x z_r$;

$\mathbf{B}_Y^{(RC)} = Q_{RC} y z_r$;

$$\mathbf{B}_Z^{(RC)} = C_5 \frac{2(a_{RC} + \xi_{RC})^2 - \rho^2}{S_{RC}^5} + B_X^{RC} \frac{\partial z_s}{\partial x} + B_Y^{RC} \frac{\partial z_s}{\partial y} - Q_{RC} D_{RC} x \frac{\partial D_{RC}}{\partial x};$$

where $Q_{RC} = 3C_5 \xi_{RC}^{-1} S_{RC}^{-5}(a_{RC}+\xi_{RC})$

$D_{RC} = D_0 + \gamma_{RC} h_{RC}(x) + \gamma_1 h_1(x)$

$h_{T,RC} = 0.5[1+x(x^2+)$

$$h_{T,RC} = 0.5[1 + x(x^2 + L_{T,RC}^2)^{-1/2}],$$
$$h_1 = 0.5\{1 - (x+16)[(x+16)^2 + 36]^{-1/2}\},$$

$C_5(Dst) = -10220+408.5 \cdot Dst$

$\mathbf{B}_{XYZ}^{(C)} = C_3(F^+_{x,y,z} + F^-_{x,y,z}) + C_4(F^+_{x,y,z} - F^-_{x,y,z})$, where

$$\begin{Bmatrix} F_x^{\pm} \\ F_y^{\pm} \end{Bmatrix} = \pm \frac{W_c(x,y)}{S^{\pm}[S^{\pm} \pm (z \pm R_T)]} \times \begin{Bmatrix} x \\ y \end{Bmatrix},$$

$$F_z^{\pm} = \frac{W_c(x,y)}{S^{\pm}} + \left(x\frac{\partial W_c}{\partial x} + y\frac{\partial W_c}{\partial y} \right) \times \frac{1}{S^{\pm} \pm (z \pm R_T)},$$

$$S^{\pm} = [(z \pm R_T)^2 + x^2 + y^2]^{1/2},$$

$$W_c(x,y) = 0.5\left[1 - \frac{x - x_{0c}}{[(x - x_{0c})^2 + L_{xc}^2]^{1/2}} \right] \times (1 + y^2/D_{yc}^2)^{-1};$$

$$\mathbf{B}_X^{(M)} = e^{x/\Delta x}[C_6 z \cos\psi + (C_7 + C_8 y^2 + C_9 z^2)\sin\psi],$$

$$\mathbf{B}_Y^{(M)} = e^{x/\Delta x}[C_{10} yz \cos\psi + (C_{11} y + C_{12} y^3 + C_{13} yz^2)\sin\psi], \text{ and}$$

$$\mathbf{B}_Z^{(M)} = e^{x/\Delta x}[(C_{14} + C_{15} y^2 + C_{16} z^2)\cos\psi + (C_{17} z + C_{18} zy^2 + C_{19} z^3)\sin\psi],$$

where C_6 through C_{19} are given by:

C_n \ Kp	$= 0,0^+$	$= 1^-,1,1^+$	$= 2^-,2,2^+$	$= 3^-,3,3^+$	$= 4^-,4,4^+$	$\geq 5^-$
C_6	1.813	2.316	2.641	3.181	3.607	4.090
C_7	31.10	35.64	42.46	47.50	51.10	49.09
C_8	-0.07464	-0.0741	-0.07611	-0.1327	-0.1006	-0.0231
C_9	-.07764	-0.1081	-0.1579	-0.1864	-0.1927	-0.1359

C_{10}	0.003303	0.003924	0.004078	0.01382	0.03353	0.01989
C_{11}	-1.129	-1.451	-1.391	-1.488	-1.392	-2.298
C_{12}	0.001663	0.00202	0.00153	0.002962	0.001594	0.004911
C_{13}	0.000988	0.00111	0.000727	0.000897	0.002439	0.003421
C_{14}	18.21	21.37	21.86	22.74	22.41	21.79
C_{15}	-0.03018	-0.04567	-0.04199	-0.04095	-0.04925	-0.05447
C_{16}	-0.03829	-0.05382	-0.06523	-0.09223	-0.1153	-0.1149
C_{17}	-0.1283	-0.1457	-0.6412	-1.059	-1.399	-0.2214
C_{18}	-0.001973	-0.002742	-0.000948	-0.001766	0.000716	-0.01355
C_{19}	0.000717	0.001244	0.0002276	0.003034	0.002696	0.001185

and where L_y=10.0, D_x=13.0, L_{RC}=5.0, L_T=6.30, γ_T=4.0, δ=0.010, γ_1=1.0, R_T=30.0, x_{0c}=4.0, L_{xc}=50.0, and D_{yc}=20.0.

Additional parameters relevant to the present invention given in the Tsyganenko reference include (from page 12 of Tsyganenko, *supra*):

Kp / parameter	$= 0,0^+$	$= 1^-,1,1^+$	$= 2^-,2,2^+$	$= 3^-,3,3^+$	$= 4^-,4,4^+$	$\geq 5^-$
N	3975	9977	9848	7309	3723	1850
<Bc>	15.49	19.06	21.71	25.48	28.58	32.88
C_1	6.51	8.52	9.75	11.35	12.41	15.12
C_2	-98.72	-35.64	-77.45	-70.12	-162.50	-128.40
C_3	-10014	-12800	-14588	-16125	-15806	-16184
C_4	15.03	14.37	64.85	90.71	160.60	149.10
C_5	76.62	124.50	123.90	38.08	5.888	215.50

Δx	24.74	22.33	20.90	18.64	18.31	19.48
a_{RC}	8.161	8.119	6.283	6.266	6.196	5.831
D_0	2.08	1.664	1.541	0.9351	0.7677	0.3325
γ_{RC}	-0.8799	0.9324	4.183	5.389	5.072	6.472
R_c	9.084	9.238	9.609	8.573	10.06	10.47
G	3.838	2.426	6.591	5.935	6.668	9.081
a_T	13.55	13.81	15.08	15.63	16.11	15.85
D_y	26.94	28.83	30.57	31.47	30.04	25.27
X_0	5.745	6.052	7.435	8.103	8.260	7.976

It should be noted that the model for $\mathbf{B}_{TSYG}$ differs from the model given in Tsyganenko, *supra*. In the first place, the author has identified several errata to this model. In the second place, the present inventors have modified the model, making C_5 a linear function of Dst for all geomagnetic activity levels. *See* Boberg et al.

Referring to FIG. 3, the Lorentz equation using a corrected model for the geomagnetic **B** field is used to calculate the geomagnetic transmission function, using the method depicted in this flowchart. Given the large computational requirements for this calculation, it is preferred to make this calculation in advance for as many orbits as desired, and storing the results as a series of geomagnetic transmission function tables, with the geomagnetic transmission probability given for each particle rigidity (rigidity ≡ momentum/charge), where each table corresponds to the geomagnetic transmission function for a particular orbit. The user may then select from several predefined

orbits having predefined geomagnetic transmission functions. Alternatively, especially as computer processing power increases, in another embodiment of the invention, this calculation of the geomagnetic transmission function may be performed after a user has specified an orbit.

Referring to **300**, the calculation of the geomagnetic transmission function will typically begin by specifying a particle rigidity in GV, a range of particle arrival directions by specifying a range of azimuth angles between θ_{max} and θ_{min} and a range of zenith angles between ϕ_{max} and ϕ_{min}, specifying an earth orbit, and specifying a geomagnetic activity level by specifying values for Kp and Dst, where Kp = [IATME or IAGA] planetary Kp index and Dst = the hourly equatorial Dst index. A standard reference for the definition of Kp is Bartels, J., "The standard index, Ks, and the planetary index, Kp", IATME Bulletin 12b, 97 IUGG Pub. Office, Paris, 1949. IATME is the International Association of Terrestrial Magnetism and Electricity. IAGA is the International Association of Geomagnetism and Aeronomy. IUGG is the International Union of Geodesy and Geophysics. A standard reference for the definition of Dst is Sugiura, M., "Hourly values of equatorial Dst for the IGY", Ann. Int. Geophys. Year, 35, 49, 1964. The Dst index was adopted by IAGA at the 1969 meeting in Madrid. IAGA Bulletin 27, 1969, 123, resolution 2. All of the references cited in this paragraph are incorporated herein by reference, in their entireties, for all purposes.

For each value of particle rigidity, the transmission probability will be calculated, and each of these results will be compiled into a table giving the geomagnetic transmission function for a given orbit. A separate geomagnetic transmission function table will be prepared for each specified

geomagnetic activity level. The user will also specify an arbitrary number of arrival directions per orbital step M_{adpos}.

Referring to **305**, the user will specify an arbitrary number of orbital steps N_{spo}, where N_{spo} is the number of steps in one orbital revolution of the satellite around the Earth's spin axis. The user will also specify a required accuracy for the numerical integration. This embodiment of the invention relies on tracing the particle trajectories backwards from a discrete number of evenly spaced satellite orbital positions. Persons of ordinary skill in the art will recognize, however, that alternative embodiments might be employed, such as embodiments using a Monte Carlo simulation, to trace particle trajectories backwards from randomly selected satellite orbital positions.

Referring to **310**, the user will specify the satellite orbit.

Referring to **315**, the software will calculate the orbital period from the orbit parameters, using standard orbit generation routines. *See* Adams et al., NRL Memorandum Report 5099, *supra*. The software will also calculate a maximum number of steps, and a time per step for the satellite to travel through a step.

Referring to **320**, the software will calculate the position of the satellite after each step. *See* Adams et al., NRL Memorandum Report 5099, *supra*. The software will also begin tracing the position of a particle incident on the satellite by initializing the particle trajectory time.

Referring to **325**, for each of the specified number of arrival directions per orbital step, the software will select a random particle arrival direction. Persons of ordinary skill in the art will

recognize, however, that alternative embodiments might be employed, such as embodiments where a number of regularly spaced arrival directions are used.

Referring to **330**, the software will calculate the particle's final position and velocity vector **v** from the satellite's latitude, longitude, altitude, and the particle's rigidity, final θ, and final ϕ.

Referring to **330, 340, 345**, the software will calculate a new particle position by performing a numerical integration of the Lorentz equation. Although this preferred embodiment uses the Bulirsch-Stoer numerical integration technique, persons of ordinary skill in the art will recognize that other numerical integration techniques are available. This numerical integration is performed on a modified Lorentz equation, where Q is replaced by $-Q$ and t' is replaced by $-t'$. As persons of ordinary skill in the art will note, the Lorentz equation is invariant if one makes this substitution. This significantly simplifies the numerical integration, because the solution in this case will represent tracing a particle backwards in time from the satellite to the boundary of the magnetosphere (for allowed trajectories) or earth's surface or atmosphere (for forbidden trajectories). This will typically entail far fewer calculations than if particles incident on the magnetosphere were traced forward in time to see if they intersected the satellite. However, persons of ordinary skill in the art will recognize that this tracing forward in time is an equally valid method.

Referring to **350, 355, 360**, the software will determine whether, after this iteration of the numerical integration, the particle's new position has placed it within the atmosphere or the solid earth (representing a forbidden trajectory), beyond the magnetosphere as given by the model (representing an allowed trajectory), or still within the magnetosphere (representing an undetermined

trajectory). If the particle is still within the magnetosphere, another iteration of the numerical integration is performed, and a new particle position determined. If the particle is not still in the magnetosphere, i.e., particle has reached either earth's surface or atmosphere (forbidden) or the outside of the magnetosphere (allowed), the software will store this result. This preferred embodiment of the software has an error trapping technique, where if the number of particle steps becomes very large, the particle is forced to an allowed or forbidden transition.

Referring to **365**, if the software has reached a solution for a given particle arrival direction at a given orbital step, the software will move on to trace the particle arriving at the next arrival direction for that orbital step, if any.

Referring to **370**, if the software has reached solutions for all the arrival directions for the satellite at a given orbital step, the software moves the satellite along to the next orbital step, if any.

Referring to **375**, after all particle tracings for all orbital steps have been performed, the results are compiled as a geomagnetic transmission function of the form $GT(R) = N_{tr}/N_{max}$, where GT is the probability of a particle being transmitted through the magnetosphere, R is particle rigidity, and N_{tr} and N_{max} are a number of particles transmitted and a total number of particles, respectively. Typically, this function will be embodied in a lookup table.

Calculating a geomagnetic transmission function for a given orbit according to the above method is an intensive process. Computing geomagnetic transmission functions for two particular orbits (the proposed space station orbit and a typical space shuttle orbit) required several days of

processor time on a RISC workstation. Shorter computation cycles could be made on a supercomputer, but supercomputer time is in chronic short supply. Accordingly, it is preferred to calculate geomagnetic transmission functions for preselected orbits in advance, and to store these geomagnetic transmission functions for later use by users. This is the approach taken by the inventors in their CREME96 software. In cases where a user specifies an orbit that is not among the precalculated orbits, the CREME96 software will use an earlier model that uses geomagnetic cutoff tables (*see* NRL Memorandum Report 5099, *supra*) rather that geomagnetic transmission functions, and warns the user of this substitution.

Alternatively, a user would specify an orbit, and if the software did not have a precalculated geomagnetic transmission function for this orbit, the software would calculate this geomagnetic transmission function. This embodiment is anticipated to become increasingly desirable as processing capacity increases.

Referring to FIGS 4, 5, 6, and 7, these plots show the geomagnetic transmission function for a common space shuttle orbit (circular orbit, 450 km altitude, 28.5° inclination) under "quiet" (FIG. 4) and "stormy" (FIG. 5) conditions, and the geomagnetic transmission function for the proposed space station orbit (circular orbit, 450 km altitude, 51.6° inclination) under "quiet" (FIG. 6) and "stormy" (FIG. 7) conditions. "Quiet" magnetospheric conditions were taken herein to be Kp = 2 and Dst = -15 nT, while "stormy" magnetospheric conditions were taken herein to be Kp ≥ 5 and Dst = -300nT. One sees that (1) transmission drops off sharply with decreasing rigidity, and (2) the differences between stormy and quiet conditions are seen most at lower rigidities.

The embodiment of the invention implemented in the CREME 96 software operates by prompting a user to specify an orbit and magnetospheric conditions, and then looking up the appropriate geomagnetic transmission function table for the orbit and magnetospheric conditions. The CREME96 software includes lookup tables corresponding to each of FIGS. 4 through 7. This table is then applied to a flux of SEU-inducing particles in the non-trapped energetic particle environment (calculated by another part of the CREME 96 software), to determine the flux of such SEU-inducing particles at the outer surface of the satellite. In other words, the flux from the non-trapped energetic particle environment will be multiplied by the probability (as embodied in the geomagnetic transmission function) that the particles will be transmitted through the magnetosphere, to get the flux at the outer surface of the satellite.

Particle Flux Modeling

Four principal classes of particles are considered to be responsible for most SEUs: galactic cosmic rays (GCRs), anomalous cosmic rays (ACRs), solar protons, and solar heavy ions. *See* NRL Memorandum Report 4506, *supra*.

To estimate the effect of these classes of particles on orbiting microelectronics, it is first necessary to estimate the exposure of the microelectronics to these particles. Thus, a feature of any complete model for the effect of cosmic rays on microelectronics will be the modeling of the flux of each of these types of particles.

In the present invention, particle fluxes are determined at the near earth environment. As used herein, the near earth environment refers to the environment found at locations approximately

1 AU from the sun, but outside of earth's magnetosphere or other localized phenomena that would perturb particle fluxes. After the particle fluxes for each type of particle is determined by the FLUX module of a preferred embodiment of the present invention, other modules may be used to model how the charged particles will induce single event upsets, including what shielding effects (including shielding by the magnetosphere and shielding by the hull of the spacecraft) will reduce this flux at the component level.

In a preferred embodiment of the invention, the flux of GCRs is modeled by the galactic cosmic ray model described by Nymmik et al. *See* "A Model of Galactic Cosmic Ray Fluxes", Nucl. Tracks Radiat. Meas. **20** 427, Nymmik et al. (1992), incorporated herein by reference, in its entirety, for all purposes. Although not essential to the present invention, the authors' updated version of this model is described at http://www.npi.msu.su/gcrf/standart/ISO_WD_15390.html, incorporated herein by reference, in its entirety, for all purposes.

In a preferred embodiment of the invention, the flux of ACRs and solar protons are modeled by the model described in Tylka et al, *supra.*

In a preferred embodiment of the invention, the flux of solar heavy ions is modeled as depicted in FIGS. 8 and 9. This routine depicts how a solar heavy ion flux for a given atomic species at a given kinetic energy, under conditions that are selected from one of three available models: worst day, worst week, and peak flux. As used herein, the worst week solar heavy ion model refers to the model based on particle fluences averaged over a 180 hour span of the solar event beginning at 1300 UT on October 19, 1989. As used herein the worst day solar heavy ion model refers to the model based on particle fluences averaged over an 18 hour span (beginning at 1300 UT on October

20, 1989). As used herein, the peak flux solar heavy ion model refers to the model based on the highest fluxes averaged over 5 minute intervals reported by the Geostationary-orbiting Operational Environmental Satellite (GOES) satellite.

Referring to **800**, this routine takes as inputs the atomic number of the selected species, and the kinetic energy for that species whose flux is to be determined. The routine also takes as inputs the user selected baseline model (worst week, worst day, or peak flux).

Referring to **805**, if the selected species has atomic number greater than 20, iron is selected as the model spectrum for the flux of this species **810**. Otherwise, **815**, oxygen is selected as the model spectrum for the flux of this species.

Referring to **820**, the next step is to look up an elemental breakpoint EB2. This breakpoint represents a value for the kinetic energy where the relationship between the flux and the kinetic energy changes to a degree where a different mathematical expression for this relationship is warranted. Values for EB2 are given in Table S1:

Table S1: Values for Elemental Breakpoint EB2, in MeV/nuc

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	24.23	19.90
oxygen	15.94	12.89

Optionally, this lookup may be performed later in the routine, after the IF step at **825**.

Referring to **825**, a special case exists for when the baseline model is the worst week model, the model element is Fe, and the kinetic energy being evaluated is above a certain cutoff point

E_{sp}=127.93 MeV/nuc. In this case, referring now to FIG. 9, values for A_{sp} and G_{sp} are looked up from Table S2 and Table S3, respectively, and the unscaled flux is calculated to be $A_{sp}\times En^{-Gsp}$, **900,910**.

Table S2: Values for A_{sp}:

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	not applicable	3.16814×10^{6}
oxygen	not applicable	not applicable

Table S3: Values for G_{sp}:

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	not applicable	2.861
oxygen	not applicable	not applicable

Referring back to FIG. 8, if En > EB2 (**835**), values for A_3 and γ_{si} are retrieved from tables S4 and S5, respectively (**840**), and the unscaled flux is calculated to be $A_3\times En^{-\gamma si}$ (**845**).

Table S4: Values for A_3

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	0.252948×10^{10}	0.249719×10^{9}
oxygen	0.106702×10^{10}	0.667628×10^{9}

Table S5: Values for γ_{s1}

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	4.52970	3.7610
oxygen	4.14060	3.76850

Otherwise, for En≤EB2, values for A_2 and G_{s1} are retrieved from tables S6 and S7, respectively (**855**), and the unscaled flux is calculated to be $A_2\exp(-G\times En^{1/4})\times En^{1/4}$ (**860**).

Table S6: Values for A_2

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	1.8991×10^{8}	3.0372×10^{8}
oxygen	$4.9518w\times10^{8}$	1.1307×10^{9}

Table S7: Values for G_{s1}

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	5.70	5.70
oxygen	5.70	5.70

For heavier species (IZ≥3), the flux must be scaled by looking up scale factors for the species being analyzed and the model element (if different from the species being analyzed). These scale factors are taken from Table S8:

Table S8: Scale factors SF for atomic species with IZ≥3

Atomic Number	Scale Factor	Atomic Number	Scale Factor	Atomic Number	Scale Factor	Atomic Number	Scale Factor	Atomic Number	Scale Factor	Atomic Number	Scale Factor
5	0	22	4.377×10^{-3}	39	4.878×10^{-6}	56	4.878×10^{-6}	73	2.195×10^{-8}	90	4.878×10^{-8}
6	4.704×10^{-1}	23	4.088	40	1.22×10^{-5}	57	4.878×10^{-7}	74	2.439×10^{-7}	91	0
7	1.2059×10^{-1}	24	1.65×10^{-2}	41	9.756×10^{-7}	58	1.22×10^{-6}	75	4.878×10^{-8}	92	2.927×10^{-8}
8	1	25	5.625×10^{-3}	42	4.878×10^{-6}	59	1.951×10^{-7}	76	7.317×10^{-7}	3	0
9	4.560976×10^{-5}	26	1	43	0	60	9.756×10^{-7}	77	7.317×10^{-7}	4	0
10	2.1312×10^{-1}	27	1.303×10^{-2}	44	2.195×10^{-6}	61	0	78	1.463×10^{-6}		
11	1.744715×10^{-2}	28	3.172×10^{-2}	45	4.878×10^{-7}	62	2.439×10^{-7}	79	2.439×10^{-7}		
12	2.0624×10^{-1}	29	3.048×10^{-4}	46	1.463×10^{-6}	63	9.756×10^{-8}	80	2.439×10^{-7}		
13	1.826829×10^{-2}	30	7.457×10^{-4}	47	4.878×10^{-7}	64	4.878×10^{-7}	81	2.195×10^{-7}		
14	3.5935×10^{-1}	31	4.878×10^{-5}	48	1.707×10^{-6}	65	7.317×10^{-8}	82	2.439×10^{-6}		
15	2.279675×10^{-4}	32	1.22×10^{-4}	49	2.195×10^{-7}	66	4.878×10^{-7}	83	1.463×10^{-7}		
16	9.758×10^{-2}	33	7.317×10^{-6}	50	4.878×10^{-6}	67	9.756×10^{-8}	84	0		
17	1.680488×10^{-4}	34	7.317×10^{-5}	51	3.415×10^{-7}	68	2.439×10^{-7}	85	0		
18	1.771545×10^{-3}	35	9.756×10^{-6}	52	7.317×10^{-6}	69	4.878×10^{-8}	86	0		
19	3.644715×10^{-4}	36	4.878×10^{-5}	53	1.463×10^{-6}	70	1.951×10^{-7}	87	0		
20	4.826×10^{-2}	37	7.317×10^{-6}	54	6.585×10^{-6}	71	4.878×10^{-8}	88	0		
21	2.929×10^{-4}	38	2.439×10^{-5}	55	4.878×10^{-7}	72	1.951×10^{-7}	89	0		

As embodied in preferred CREME96 software according to the invention, this routine is run for each atomic species in the solar heavy ion flux, and for each value of kinetic energy in the solar

heavy ion flux. To determine the total solar heavy ion flux, the fluxes for each kinetic energy of each species are added together.

Internet Implementation

The preferred CREME96 software embodiment of the present invention is large and complex. Moreover, it is anticipated that as models used in the software are improved and/or updated, it will be necessary to perform maintenance on this software. Accordingly, as noted *supra*, a preferred embodiment of the invention includes a network interface for allowing remote users to access and run the software. Since the internet is a stateless system, a user does not maintain a connection to the server as users in a client-server environment would. One of the goals of the internet implementation of the present invention was to provide users with as much of the functionality of a client-server based system as possible in the stateless system of the internet.

The preferred system for allowing a large number of users to remotely access and run software such as CREME96 is a system connected to the internet, or equivalently a system connected to an intranet. Accordingly, the preferred embodiment of the invention is a server computer (or equivalently, a plurality of server computers connected in parallel) with the CREME96 software, with PERL script software (or equivalently Java scripts, or scripts written in some other script language) for controlling input to and output from the CREME96 software, as well as other functions, and with listener software and other hardware and software necessary to permit remote users with internet access to access the CREME96 software through their respective web browsers. Listings for the preferred software according to the invention are found *infra*.

Referring to FIG. 10, this figure shows a main menu for the preferred CREME96 software according to the invention. This menu may be accessed through the CREME96 web site maintained at the Naval Research Laboratory having a URL http://crsp3.nrl.navy.mil/creme96/. This page is generated by a PERL script in response to a registered user transmitting to the server a valid username and password. The PERL script will compare the transmitted username and password to a list stored on the server, and if a matching entry is found, the script will generate a CREME96 main menu page customized for that user.

One of the advantages of CREME96 over other web-based applications is that it stores information for each user on the server. This obviates the need for cookies or other downloads to the user's computer. Many users find the downloading of cookies to be objectionable. Moreover, keeping this information on the server may speed up overall operation, because there is no need for the server to download this information to the user, and the user to subsequently upload it to the server. Thus, when a user enters a valid username and password, the script will search a section of the hard drive set aside for that user for any User Request Files (URFs). The user will then be able to pick any of the user's previously stored URFs to use as a data file for a selected routine.

The user has the option, from this main menu, of either running one or more modeling routines, using a selected URF, editing or creating an URF, or running utilities.

The available routines include the GTRN routine for calculating a geomagnetic transmission function as described *supra*, the FLUX routine for calculating a flux of SEU-inducing particles, as described *supra*, the TRANS routine for calculating the transmission of SEU-inducing particles through solid shielding, as described *supra*, the LETSPEC routine for calculating the linear energy

transfer of charged particles to electronic devices, as described *supra*, and the PUP and HUP routines for calculating the SEU rates for electronic devices exposed to SEU-inducing particles.

The user will select routines to run by clicking a check box next to each of the routines to be run, and picking an URF for each routine to be run from the list of URFs that user has previously stored for that routine. When the user sends this message to the server, by pressing the GO button, a script will generate a system command to run each of these selected routines, using the selected URF as an input file for that routine. As persons of ordinary skill in the art will recognize, a system command is a computer command at the system level. In this case, the system command is to run a program with specified inputs.

Alternatively, if the user selects a radio button next to one of these routines and clicks the GO button, this message from the user will trigger a script to generate and transmit to the user an HTML page with inputs for the data fields in the corresponding URF. If the user has first specified an existing (i.e., stored on the server) URF, the values for the fields in this URF will be included in the generated HTML page, allowing the user to edit the values for the fields in the URF. The URF file name is included in the URF fields, thereby providing the convenience of creating a new URF derived from editing a previous URF, and allowing file storage of both the original and modified URF.

One of the features of the preferred embodiment of the invention is the connectivity between the routines. Referring back to FIG. 1, one sees that the modules of the software are typically executed in the indicated order. In the preferred embodiment, the modules receive input in the form of a URF. The modules also receive input from a file generated by the preceding module (this file

specified by the URF), and/or send output to a file for use by the subsequent module (this file likewise specified by the URF). For instance, the GTRN module will generate a .GT* output file. The FLUX module may use this .GT* file as an input, and will output a .FLX output file. The TRANS module will typically use a .FLX file as an input, and output a .TFX file. The LETSPEC module will typically use a .TFX file (or optionally a .FLX file) as an input, and will output a .LET file. The PUP module will typically use a .TFX file (or optionally a .FLX file) as an input, and will output a .PUP proton-induced upset report file. The HUP module will use a .LET file as an input, and will output a .HUP direct ionization induced upset report file.

Accordingly, by specifying a URF for each of these routines, selecting each of these routines to run, and clicking the GO button, a user will send a message to the server that will cause the scripts to run each of these routines in order, using the outputs from the preceding routines and generating inputs for the subsequent routines. The result of these calculations will be estimates of the single event upset rates for the specified electronics in the specified orbit under the specified conditions. The scripts will generate and transmit an HTML page with these results to the user, who may then use the utilities to download the results, create plots, or perform other functions.

Having described the invention, the following examples are given to illustrate specific applications of the invention, including the best mode now known to perform the invention. These specific examples are not intended to limit the scope of the invention described in this application.

Example 1:
CREME96

Code for CREME96, a preferred embodiment of the present invention, follows. The software may be accessed at its internet site with URL http://crsp3.nrl.navy.mil/creme96/.

Obviously, many modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that, within the scope of the appended claims, the invention may be practiced otherwise than as specifically described.

CLAIMS

What is claimed is:

1. A method for determining a geomagnetic transmission function for the transmission of a population of particles, said particles having one or more rigidities, to a satellite in a known earth orbit, under known geomagnetic conditions, comprising the steps:

(A) dividing said orbit into a plurality of sequential steps comprising at least a first step and a last step, each sequential pair of said steps being connected at a point, each of said points corresponding to a position of said satellite along said orbit, and specifying a value for Kp and a value for Dst to specify said geomagnetic conditions;

(B) for a plurality of said points, selecting a plurality of arrival directions for the arrival of said particles at said satellites, wherein each of said arrival directions represents either an allowed or a forbidden trajectory of one of said particles; and

(C) for each of said arrival directions at each of said plurality of said points, determining whether said arrival direction represents an allowed or a forbidden trajectory by tracing a path of said particle to arrive at said point with said arrival direction, until said particle path intersects a boundary of earth's magnetosphere, thereby indicating an allowed trajectory, or until said particle path intersects either earth's atmosphere or earth's surface, thereby indicating a forbidden trajectory, wherein said tracing of said path is performed by integrating in the time domain the Lorentz equation

$$\mathbf{F} = m\gamma\, d\mathbf{v}/d(-t') = -Q\mathbf{v}\times\mathbf{B}$$

wherein $\mathbf{F}$ is the force vector acting on said particle, m is mass, γ is $1/(1-\mathbf{v}^2/c^2)^{-1/2}$, $\mathbf{v}$ is said particle's velocity vector, t' is the travel time of said particle, Q is the charge of said particle, and $\mathbf{B}$ is the magnetic field vector acting on said particle, wherein $\mathbf{B}$ is given by

$$\mathbf{B} = \mathbf{B}_{IGRF}(\mathbf{r}, t') + \mathbf{B}_{TSYG}(Kp, \mathbf{r}, t')$$

wherein $\mathbf{B}_{IGRF}$ is the International Geomagnetic Reference Field promulgated by the International Association of Geomagnetism and Aeronomy, and $\mathbf{B}_{TSYG}$ is the modified Tsyganenko field given by the sum of $\mathbf{B}_{Xmg}{}^{(T)} + \mathbf{B}_{Ymg}{}^{(T)} + \mathbf{B}_{Zmg}{}^{(T)} + \mathbf{B}_{Xmg}{}^{(RC)} + \mathbf{B}_{Ymg}{}^{(RC)} + \mathbf{B}_{Zmg}{}^{(RC)} + \mathbf{B}_{Xsm}{}^{(C)} + \mathbf{B}_{Ysm}{}^{(C)} + \mathbf{B}_{Zsm}{}^{(C)} + \mathbf{B}_{Xsm}{}^{(M)} + \mathbf{B}_{Ysm}{}^{(M)} + \mathbf{B}_{Zsm}{}^{(M)}$, wherein coordinates in the solar magnetospheric system are denoted sm, and wherein coordinates in the solar magnetic coordinate system are denoted mg; and wherein

$\mathbf{B}_X{}^{(T)} = Q_t\, x\, z_T;$

$\mathbf{B}_Y{}^{(T)} = Q_t\, y\, z_T;$

$$\mathbf{B}_z(T) = \frac{W(x,y)}{S_T}\left(C_1 + C_2\frac{a_T+\xi_T}{S_T^2}\right) + \frac{x\frac{\partial W}{\partial x} + y\frac{\partial W}{\partial y}}{S_T + a_T + \xi_T}\times (C_1 + C_2/S_T) + \mathbf{B}_X(T)\frac{\partial z_s}{\partial x} + \mathbf{B}_Y(T)\frac{\partial z_s}{\partial y} - Q_T D_T\left(x\frac{\partial D_T}{\partial x} + y\frac{\partial D_T}{\partial y}\right)$$

wherein $W(x,y) = 0.5\left(1 - \frac{x - x_0}{\left[(x - x_0)^2 + D_x^2\right]^{1/2}}\right) \times \left(1 + y^2 / D_y^2\right)^{-1}$

wherein $Q_T = \frac{W(x,y)}{\xi_T S_T}\left[\frac{C_1}{S_T + a_T + x_T} + \frac{C_2}{S_T^2}\right]$;

wherein $z_r = z - z_s(x, y, \psi)$,

$$z_s(x,y,\psi) = 0.5 \tan\psi\left(x + R_c - \sqrt{(x + R_c)^2 + 16}\right) - G\sin\psi \cdot y^4\left(y^4 + L_y^4\right)^{-1},$$

wherein

$$S_{T,RC} = \sqrt{\rho^2 + \left(a_{T,RC} + \xi_{T,RC}\right)^2},$$

$$\xi_{T,RC} = \sqrt{z_r^2 + D_{T,RC}^2},$$

$$D_T = D_0 + \delta y^2 + \gamma_T h_T(x) + \gamma_1 h_1(x),$$

wherein $C_1 = -98.72$ when $Kp = 0, 0^+$, $C_1 = -35.64$ when $Kp = 1^-, 1, 1^+$, $C_1 = -77.45$ when $Kp = 2^-, 2, 2^+$, $C_1 = -70.12$ when $Kp = 3^-, 3, 3^+$, $C_1 = -162.5$ when $Kp = 4^-, 4, 4^+$,

$C_1 = -128.4$ when $Kp \geq 5^-$,

wherein $C_2 = -10014$ when $Kp = 0, 0^+$, $C_2 = -12800$ when $Kp = 1^-,1,1^+$, $C_2 = -14588$ when $Kp = 2^-,2,2^+$, $C_2 = -16125$ when $Kp = 3^-,3,3^+$, $C_2 = -15806$ when $Kp = 4^-,4,4^+$, $C_2 = -16184$ when $Kp \geq 5^-$,

wherein $a_T = 13.55$ when $Kp = 0, 0^+$, $a_T = 13.81$ when $Kp = 1^-,1,1^+$, $a_T = 15.08$ when $Kp = 2^-,2,2^+$, $a_T = 15.63$ when $Kp = 3^-,3,3^+$, $a_T = 16.11$ when $Kp = 4^-,4,4^+$, $a_T = 15.85$ when $Kp \geq 5^-$,

wherein $D_0 = 2.08$ when $Kp = 0, 0^+$, $D_0 = 1.664$ when $Kp = 1^-,1,1^+$, $D_0 = 1.541$ when $Kp = 2^-,2,2^+$, $D_0 = 0.9351$ when $Kp = 3^-,3,3^+$, $D_0 = 0.7677$ when $Kp = 4^-,4,4^+$, $D_0 = 0.3325$ when $Kp \geq 5^-$,

wherein $R_c = 9.084$ when $Kp = 0, 0^+$, $R_c = 9.238$ when $Kp = 1^-,1,1^+$, $R_c = 9.609$, when $Kp = 2^-,2,2^+$, $R_c = 8.573$ when $Kp = 3^-,3,3^+$, $R_c = 10.06$ when $Kp = 4^-,4,4^+$, $R_c = 10.47$ when $Kp \geq 5^-$,

wherein $L_y = 10\ R_E$,

$\mathbf{B}_X^{(RC)} = Q_{RC} x z_r$;

$\mathbf{B}_Y^{(RC)} = Q_{RC} y z_r$;

$$\mathbf{B}_Z^{(RC)} = C_5 \frac{2(a_{RC} + \xi_{RC})^2 - \rho^2}{S_{RC}^5} + B_X^{RC} \frac{\partial z_s}{\partial x} + B_Y^{RC} \frac{\partial z_s}{\partial y} - Q_{RC} D_{RC} x \frac{\partial D_{RC}}{\partial x};$$

wherein $Q_{RC} = 3C_5 \xi_{RC}^{-1} S_{RC}^{-5} (a_{RC} + \xi_{RC})$

$$D_{RC} = D_0 + \gamma_{RC} h_{RC}(x) + \gamma_1 h_1(x)$$

$$h_{T,RC} = 0.5[1 + x(x^2 + L^2_{T.RC})^{-1/2}],$$
$$h_1 = 0.5\{1 - (x + 16)[(x + 16)^2 + 36]^{-1/2}\},$$

$$C_5(Dst) = -10220 + 408.5 \cdot Dst$$

$\mathbf{B}_{XYZ}{}^{(C)} = C_3(F^{+}{}_{x,y,z} + F^{-}{}_{x,y,z}) + C_4(F^{+}{}_{x,y,z} - F^{-}{}_{x,y,z})$, wherein

$$\left\{\frac{F_x^{\pm}}{F_y^{\pm}}\right\} = \pm \frac{W_c(x, y)}{S^{\pm}[S^{\pm} \pm (z \pm R_T)]} \times \left\{\frac{x}{y}\right\},$$
$$F_z^{\pm} = \frac{W_c(x, y)}{S^{\pm}} + \left(x\frac{\partial W_c}{\partial x} + y\frac{\partial W_c}{\partial y}\right) \times \frac{1}{S^{\pm} \pm (z \pm R_T)},$$
$$S^{\pm} = [(z \pm R_T)^2 + x^2 + y^2]^{1/2},$$
$$W_c(x, y) = 0.5\left[1 - \frac{x - x_{0c}}{[(x - x_{0c})^2 + L_{xc}^2]^{1/2}}\right] \times (1 + y^2/D_{yc}^2)^{-1};$$

$$\mathbf{B}_X{}^{(M)} = e^{x/\Delta x}[C_6 z \cos\psi + (C_7 + C_8 y^2 + C_9 z^2) \sin\psi],$$

$\mathbf{B}_Y{}^{(M)} = e^{x/\Delta x}[C_{10} yz \cos\psi + (C_{11} y + C_{12} y^3 + C_{13} yz^2) \sin\psi]$, and

$$\mathbf{B}_Z{}^{(M)} = e^{x/\Delta x}[(C_{14} + C_{15} y^2 + C_{16} z^2) \cos\psi + (C_{17} z + C_{18} zy^2 + C_{19} z^3) \sin\psi],$$

wherein C_6 through C_{19} are given by:

C_n \ Kp	$= 0,0^+$	$= 1^-,1,1^+$	$= 2^-,2,2^+$	$= 3^-,3,3^+$	$= 4^-,4,4^+$	$\geq 5^-$
C_6	1.813	2.316	2.641	3.181	3.607	4.090
C_7	31.10	35.64	42.46	47.50	51.10	49.09
C_8	-0.07464	-0.0741	-0.07611	-0.1327	-0.1006	-0.0231
C_9	-.07764	-0.1081	-0.1579	-0.1864	-0.1927	-0.1359
C_{10}	0.003303	0.003924	0.004078	0.01382	0.03353	0.01989
C_{11}	-1.129	-1.451	-1.391	-1.488	-1.392	-2.298
C_{12}	0.001663	0.00202	0.00153	0.002962	0.001594	0.004911
C_{13}	0.000988	0.00111	0.000727	0.000897	0.002439	0.003421
C_{14}	18.21	21.37	21.86	22.74	22.41	21.79
C_{15}	-0.03018	-0.04567	-0.04199	-0.04095	-0.04925	-0.05447
C_{16}	-0.03829	-0.05382	-0.06523	-0.09223	-0.1153	-0.1149
C_{17}	-0.1283	-0.1457	-0.6412	-1.059	-1.399	-0.2214
C_{18}	-0.001973	-0.002742	-0.000948	-0.001766	0.000716	-0.01355
C_{19}	0.000717	0.001244	0.0002276	0.003034	0.002696	0.001185

and wherein L_y=10.0, D_x=13.0, L_{RC}=5.0, L_T=6.30, γ_T=4.0, δ=0.010, γ_1=1.0, R_T=30.0, x_{0c}=4.0, L_{xc}^2=50.0, and D_{yc}=20.0,

and thereby determining whether said particle's trajectory intersects either the boundary of earth's magnetosphere, thereby indicating an allowed trajectory, or intersecting earth's surface or earth's atmosphere, thereby indicating a forbidden trajectory.

2. The method of claim 1, wherein said plurality of arrival directions comprises one or more randomly or pseudorandomly selected arrival directions.

3. The method of claim 1, further comprising the step:

(D) for each particle rigidity, determine what fraction of particles of that rigidity will be transmitted to said satellite.

4. The method of claim 1, wherein said plurality of points comprises a complete set of points for said orbit.

5. The method of claim 1, wherein said plurality of points comprises a set of points for a portion of said orbit.

6. A method for determining, for a given particle environment outside of earth's magnetosphere, what portion of a population of particles having one or more rigidities making up said particle environment will be transmitted to a satellite in a known earth orbit, comprising the steps:

(A) performing steps (A) through (C) of claim 1, thereby computing a geomagnetic transmission function for said population of particles; and

(B) applying said geomagnetic transmission function to said population of particles, thereby determining what portion of that population of particles will be transmitted to said satellite.

7. A method for determining, for a given particle environment outside of earth's magnetosphere, what portion of a population of particles having one or more rigidities making up said particle environment will be transmitted to a satellite in earth orbit, comprising the steps:

(A) prompting a user to specify an earth orbit;

(B) determining whether said orbit is among a group of preselected orbits, each of said orbits in said preselected group of orbits having an associated predetermined geomagnetic transmission function for a range of particle rigidities, each of said predetermined geomagnetic transmission functions having been prepared in accordance with claim 1; and

(C) for the case wherein said orbit is among said preselected group of orbits, applying said predetermined geomagnetic transmission function for said orbit to said particle environment outside earth's magnetosphere.

8. The method of claim 7, wherein said preselected group of orbits comprises a quiet shuttle orbit having a 450 km altitude and a 28.5° inclination, a disturbed shuttle orbit having a 450 km altitude and a 28.5° inclination, a quiet space station orbit having a 450 km altitude and a 51.6° inclination, and a disturbed space station orbit having a 450 km altitude and a 51.6° inclination, wherein said quiet shuttle orbit has a geomagnetic transmission function given by FIG. 4, wherein said disturbed shuttle orbit has a geomagnetic transmission function given by FIG. 5, wherein said quiet space station orbit has a geomagnetic transmission function given by FIG. 6, and wherein said disturbed space station orbit has a geomagnetic transmission function given by FIG. 7.

9. The method of claim 7, further comprising the step:

(D) for the case wherein said orbit is not among said preselected group of orbits, performing a step selected from the group consisting of (a) returning an error message to a user, and (b) computing a geomagnetic transmission function for said orbit, in accordance with the method of claim 1.

10. A method for determining a flux of a species of solar ions having a specified atomic number between 3 and 92, and a specified kinetic energy, for a satellite in a near earth orbit, comprising the steps:

(A) specifying an atomic number for solar ions for evaluation;

(B) specifying a kinetic energy for said solar ions;

(C) specifying a baseline model for said flux of solar ions, wherein said baseline model is selected from the group consisting of a worst day model, a worst week model, and a peak flux model;

(D) in the case wherein said specified atomic number is greater than 20, selecting iron as an elemental spectrum model;

(E) in the case wherein said specified atomic number is less than or equal to 20, selecting oxygen as an elemental spectrum model;

(F) looking up a value for an elemental breakpoint, wherein said elemental breakpoint is a function of said elemental spectrum model and said baseline model, and wherein said elemental breakpoint is selected from the table:

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	24.23 MeV/nuc	19.90 MeV/nuc
oxygen	15.94 MeV/nuc	12.89 MeV/nuc

(G) in the case wherein said baseline model is said worst week model, and said elemental spectrum model is iron, and said kinetic energy is greater than 127.93 MeV/nuc, calculating an unscaled flux, wherein said unscaled flux equals $A_{sp} \times \left(En/{}^{MeV}\!/_{nuc}\right)^{-Gsp}$, wherein $A_{sp} = 3.16814 \times 10^{6}$

(cm^2 sr MeV/nuc)$^{-1}$, $\left(En/{}^{MeV}\!/_{nuc}\right)$ is said kinetic energy, normalized to be dimensionless, and G_{sp} = 2.861;

(H) in the case wherein the condition recited in step (G) is not satisfied, and wherein said kinetic energy is greater than said elemental breakpoint, calculating an unscaled flux wherein said unscaled flux equals $A_3 \times EN^{\gamma_{S1}}$, wherein A_3 is a function of said elemental spectrum model and said baseline model, and wherein said A_3 is selected from the table:

Elemental spectrum model \ Baseline model	worst day or peak flux in (cm^2 sr MeV/nuc)$^{-1}$	worst week in (cm^2 sr MeV/nuc)$^{-1}$
iron	0.252948×10^{10}	0.249719×10^{9}
oxygen	0.106702×10^{10}	0.667628×10^{9}

and wherein γ_{S1} is a spectral index and is a function of said elemental spectrum model and said baseline model, and wherein said γ_{S1} is selected from the table:

Elemental spectrum model \ Baseline model	worst day or peak flux	worst week
iron	-4.52970	-3.7610
oxygen	-4.14060	-3.76850

;

(I) in the case wherein the condition recited in step (G) is not satisfied, and wherein said kinetic energy is less than or equal to said elemental breakpoint, calculating an unscaled flux wherein said unscaled flux equals $A_2 \exp(-G \times En^{1/4}) \times En^{1/4}$, wherein A_2 is a function of said elemental spectrum model and said baseline model, and wherein said A_2 is selected from the table:

Elemental spectrum model \ Baseline model	worst day or peak flux in (cm^2 sr MeV/nuc)$^{-1}$	worst week in (cm^2 sr MeV/nuc)$^{-1}$

iron	1.8991×10^8	3.0372×10^8
oxygen	4.9518×10^8	1.1307×10^9

and wherein G_{s1} is a spectral index and is a function of said elemental spectrum model and said baseline model, and wherein said G_{s1} is selected from the table:

Baseline model / Elemental spectrum model	worst day or peak flux	worst week
iron	5.70	5.70
oxygen	5.70	5.70

; and

(J) in the case where said atomic number is between 3 and 92, inclusive, calculating a solar ion flux for said specified atomic number and said specified kinetic energy by multiplying said unscaled flux by a scale factor ratio, said scale factor ratio being the ratio of a scale factor for an element having said selected atomic number over a scale factor for said spectrum model element, wherein said scale factors are selected from the table:

Atomic Number	Scale Factor	Atomic Number	Scale Factor	Atomic Number	Scale Factor	Atomic Number	Scale Factor	Atomic Number	Scale Factor	Atomic Number	Scale Factor
5	0	22	4 377 $\times 10^{-3}$	39	4 878 $\times 10^{-6}$	56	4 878 $\times 10^{-6}$	73	2.195 $\times 10^{-8}$	90	4.878 $\times 10^{-8}$
6	4 704 $\times 10^{-1}$	23	4.088	40	1.22 $\times 10^{-5}$	57	4.878 $\times 10^{-7}$	74	2.439 $\times 10^{-7}$	91	0
7	1.2059 $\times 10^{-1}$	24	1.65 $\times 10^{-2}$	41	9.756 $\times 10^{-7}$	58	1.22 $\times 10^{-6}$	75	4.878 $\times 10^{-8}$	92	2.927 $\times 10^{-8}$
8	1	25	5 625 $\times 10^{-3}$	42	4.878 $\times 10^{-6}$	59	1 951 $\times 10^{-7}$	76	7.317 $\times 10^{-7}$	3	0
9	4.560976 $\times 10^{-5}$	26	1	43	0	60	9 756 $\times 10^{-7}$	77	7.317 $\times 10^{-7}$	4	0
10	2.1312 $\times 10^{-1}$	27	1.303 $\times 10^{-2}$	44	2.195 $\times 10^{-6}$	61	0	78	1.463 $\times 10^{-6}$		
11	1.744715 $\times 10^{-2}$	28	3.172 $\times 10^{-2}$	45	4 878 $\times 10^{-7}$	62	2.439 $\times 10^{-7}$	79	2.439 $\times 10^{-7}$		
12	2.0624 $\times 10^{-1}$	29	3.048 $\times 10^{-4}$	46	1.463 $\times 10^{-6}$	63	9 756 $\times 10^{-8}$	80	2.439 $\times 10^{-7}$		
13	1.826829 $\times 10^{-2}$	30	7.457 $\times 10^{-4}$	47	4 878 $\times 10^{-7}$	64	4 878 $\times 10^{-7}$	81	2.195 $\times 10^{-7}$		
14	3 5935 $\times 10^{-1}$	31	4.878 $\times 10^{-5}$	48	1.707 $\times 10^{-6}$	65	7 317 $\times 10^{-8}$	82	2.439 $\times 10^{-6}$		
15	2.279675 $\times 10^{-4}$	32	1 22 $\times 10^{-4}$	49	2 195 $\times 10^{-7}$	66	4.878 $\times 10^{-7}$	83	1.463 $\times 10^{-7}$		
16	9.758 $\times 10^{-2}$	33	7.317 $\times 10^{-6}$	50	4.878 $\times 10^{-6}$	67	9.756 $\times 10^{-8}$	84	0		
17	1 680488 $\times 10^{-4}$	34	7 317 $\times 10^{-5}$	51	3.415 $\times 10^{-7}$	68	2 439 $\times 10^{-7}$	85	0		
18	1 771545 $\times 10^{-3}$	35	9.756 $\times 10^{-6}$	52	7 317 $\times 10^{-6}$	69	4 878 $\times 10^{-8}$	86	0		
19	3.644715 $\times 10^{-4}$	36	4.878 $\times 10^{-5}$	53	1.463 $\times 10^{-6}$	70	1.951 $\times 10^{-7}$	87	0		
20	4.826 $\times 10^{-2}$	37	7.317 $\times 10^{-6}$	54	6.585 $\times 10^{-6}$	71	4.878 $\times 10^{-8}$	88	0		
21	2.929 $\times 10^{-4}$	38	2.439 $\times 10^{-5}$	55	4.878 $\times 10^{-7}$	72	1.951 $\times 10^{-7}$	89	0		

wherein elements having atomic number of 20 or less are scaled to oxygen, and elements having higher atomic numbers are scaled to iron.

11. A method for determining a flux of solar ions of species having a range of atomic numbers between 3 and 92, wherein each of said species includes particles having a range of specified kinetic energies, for a satellite in a near earth orbit, comprising the steps:

(A) for each kinetic energy of each of said species, performing steps (A) through (K) of claim 10, and storing a flux value for each of said kinetic energies for each of said species; and

(B) adding each of said flux values to obtain a total solar ion flux value.

12. A method for modeling the effects of cosmic rays on microelectronics on a computer connected to the internet, comprising the steps:

(A) receiving a login message from a remote user connected to the internet, and generating and transmitting back to said user with a script in response thereto an HTML main menu page, wherein said main menu page comprises prompts for said user to run one or more routines selected from the group consisting of: (1) a routine for calculating a geomagnetic transmission function for SEU-inducing particles, (2) a routine for calculating a flux of SEU-inducing particles in the near earth environment or in the environment shielded by earth's magnetosphere, (3) a routine for calculating a solid shielding transport function for SEU-inducing particles, (4) a routine for calculating a proton induced single event upset rate, (5) a routine for calculating a linear energy transfer rate, and (6) a routine for calculating a heavy ion induced single event upset rate, said main menu page further comprises prompts for said user to select a user request file for each of said routines;

(B) in response to a message from said user to run at least one of said routines, using a user request file specified by said user, generating and executing with a script a system command for said

computer to run each of said specified routines; and

(C) in response to the completion of all of said routines specified by said user, generating and transmitting to said user with a script an HTML page notifying said user of said completion.

13. The method of claim 12, wherein said main menu page generated and transmitted to said user further comprises prompts for said user to create or edit a user request file for any one of said routines, in lieu of running said one or more of said routines, and further comprising the step:

(D) in response to a message from said user to create or edit a user request file, generating and transmitting to said user with a script an HTML page with prompts for the entry of fields for said user request file; and

(E) in response to said user transmitting information for said user request file, storing said information in a user request file with a script.

14. The method of claim 12, wherein said main menu page comprises prompts for said user to run a routine for calculating a geomagnetic transmission function for SEU-inducing particles, and wherein said step of generating and executing with a script a system command for said computer to run each of said specified routines comprises at least generating and executing a system command for said computer to run said routine for calculating a geomagnetic transmission function for SEU-inducing particles, in accordance with claim 1

15. The method of claim 12, wherein said main menu page comprises prompts for said user to run a routine for calculating a flux of SEU-inducing particles in the near earth environment or in the

environment shielded by earth's magnetosphere, and wherein said step of generating and executing with a script a system command for said computer to run each of said specified routines comprises at least generating and executing a system command for said computer to run said routine for calculating said flux of SEU-inducing particles in the near earth environment or in the environment shielded by earth's magnetosphere, wherein said routine comprises calculating a flux of solar heavy ions in accordance with claim 10.

ABSTRACT OF THE DISCLOSURE

An aspect of the present invention is a method and apparatus for computing a geomagnetic transmission function. This apparatus includes a programmed digital computer running modeling software for modeling the transmission of cosmic ray particles through the magnetosphere. The software includes a model representing a solution to the Lorentz equation in a magnetic field given by $\mathbf{B} = \mathbf{B}_{IGRF}(\mathbf{r},t') + \mathbf{B}_{TSYG}(Kp,\mathbf{r},t')$. Another aspect of the present invention is a method and apparatus for computing a flux of particles at the outer surface of a satellite comprising, *inter alia*, an improved method and apparatus for computing a flux of solar heavy ions. This apparatus includes a programmed digital computer running modeling software for modeling the flux of cosmic ray particles through the outer surface of a satellite. Another aspect of the invention is a method and apparatus comprising a programmed digital computer running modeling software for modeling the effect of cosmic rays on microelectronics, where this software embodies at least one of the two foregoing aspects of the invention. Another aspect of the invention is a preferred embodiment of a method and apparatus comprising a programmed digital computer running modeling software for modeling the effect of cosmic rays on microelectronics, where this software embodies at least one of the two foregoing aspects of the invention, where this preferred embodiment is connected to a network, typically the internet, to permit remote users to use the invention.

```fortran
      REAL FUNCTION ACR_FLUX(IZ,IQ,EN,YEAR)
C
C     Returns the interplanetary anomalous cosmic ray (ACR) flux for element
C     IZ in charge state IQ at energy EN (in MeV/nuc) for year YEAR
C     (ie, decimal year 1987.23)
C
C     Flux returned in unit of ions/m2-s-sr-MeV/nuc
C
C      Written by:    Allan J. Tylka
C                     Code 7654
C                     Naval Research Laboratory
C                     Washington, DC 20375-5352
C                     tylka@crs2.nrl.navy.mil
C
C      Last update:   25 September 1996
C
C-----------------------------------------------------------------------------
C
C     This routine based on results from several sources:
C
C     (1) ACR Elemental Composition
C      Cummings & Stone discovered that all ACR spectra could be mapped
C      to a common spectral shape by using flux and energy scaling factors,
C      contained here in the arrays FFAC and EFAC, respectively.  The values
C      of these scaling factors are taken from various sources.  Unless
C      otherwise noted, they are given in:
C      Cummings & Stone, Proc. 20th ICRC (Moscow), 3, 413-416 (1987).

C      FFAC values for N, Ne modified to match the interplanetary abundances
C      reported from SAMPEX at 16-25 MeV/nuc:
C      Selesnick et al., JGR 100, 9503-9518 (1995).
C
C      He spectrum taken from:
C      Cummings, Stone, & Webber, Ap.J. Lett. 287, L199-L103 (1984), which
C      is based on Voyager observations in 9/1977-2/1978, when the
C      spacecraft were still near Earth (~1.3-2.5 AU)
C
C      Relative abundance and spectrum of ACR H at 1 AU estimated by:
C      Mewaldt, Proc. 24th ICRC (Rome) 4, 808-811 (1995).
C      (The spectrum here roughly matches this.)
C
C     (2) Spectral shape and distribution of charge states:
C
C      Based on SAMPEX observations and modeling results presented by
C      R.A. Mewaldt et al., Astrophysical Journal Letters 466, L43-L46 (1996).
C
C     (3) Temporal variation and solar-cycle modulation:
C
C      Based on the timeline of 8-27 MeV/nuc oxygen for 1968-1994 as reported
C      by  Mewaldt et al. GRL 20, 2263-2266 (1993) and augmented during the
C      1985-95 time period by Cosmos measurements reported by: Beaujean et al.
C       Proc. 24th ICRC (Rome) 4, 832-835 (1994).

      IMPLICIT NONE
      INTEGER*4 IZ,IQ,NELM
      PARAMETER (NELM=18)
      REAL*4 EN,YEAR,FPEAK,AMASS,FFAC,EFAC,EN1
      DIMENSION FPEAK(4),FFAC(NELM),EFAC(NELM),AMASS(NELM)
      DATA FPEAK/3.935,0.3808,0.1014,3.01E-2/
      DATA FFAC/3.6,4.9,3*0.0,0.0075,0.127,1.0,0.0,0.40,7*0.0,0.019/
```

```
      DATA EFAC/9.0,2.6,3*1.0,1.38,1.14,1.0,1.0,0.64,7*1.0,0.36/
      DATA AMASS/
     & 1.00794,4.002602,6.941,9.012182,10.811,12.011,14.00674,15.9994,
     & 18.9984032,20.1797,22.989768,24.305,26.981539,28.0855,30.973762,
     & 32.066,35.4527,39.948/
      REAL*4 A0,Q,AN,FP,BETA,ANORM,EPEAK
      DATA A0/1.70/
      REAL*4 SMNORM/0.5534E+4/
      REAL*4 ACRO_TIMELINE
C
      ACR_FLUX=0.0
      IF (IZ.GT.NELM) RETURN
      IF (FFAC(IZ).LE.0) RETURN
      IF (EN.LT.1.0 .or. EN.GT.1000.) RETURN
      IF (IQ.GT.IZ) RETURN

      Q=IQ*1.0
      AN=AMASS(IZ)
      IF (IQ.LE.4) THEN
          FP=FPEAK(IQ)
      ELSEIF (IQ.GT.4) THEN
          FP=3.935/Q**3.52
      ENDIF
      EPEAK=6.73*Q**0.91
      FP=FP*FFAC(IZ)
      BETA=1.0/A0/EPEAK**A0
      ANORM=SMNORM*FP/EPEAK/EXP(-BETA*EPEAK**A0)
      EN1=EN/EFAC(IZ)
      ACR_FLUX=ANORM*EN1*EXP(-BETA*EN1**A0)

C     Solar modulation factor:
      ACR_FLUX=ACR_FLUX*ACRO_TIMELINE(YEAR)
      RETURN
      END

      REAL FUNCTION ACRO_TIMELINE(USERX)
C
C     Function to model the solar cycle variation of the anomalous
C     component at 1 AU.  Based on the timeline of 8-27 MeV/nuc oxygen
C     for 1968-1994 as reported by  Mewaldt et al. GRL 20, 2263-2266 (1993)
C     and augmented during the 1985-95 time period by Cosmos measurements
C     reported by: Beaujean et al. Proc. 24th ICRC (Rome) 4, 832-835 (1994).
C
C Calculates a y-value, along a line drawn through data points, for a given
C year (USERX) on Fig.3 of Mewaldt et al.

      IMPLICIT NONE
C Argument declarations
      REAL USERX

C Local declarations
      INTEGER KMAX,I
      PARAMETER (KMAX=5)
      REAL XVAL(KMAX+1),YVAL(KMAX+1),SLOPE(KMAX),YEAR
      DATA XVAL /1.9685E+03,1.9715E+03,1.9778E+03,1.9825E+03,
     & 1.9873E+03,1.9903E+03/
      DATA YVAL /5.8651E-09,1.2123E-06,2.0350E-06,6.4463E-09,
     & 1.7630E-06,5.8651E-09/
      DATA SLOPE /1.782245,8.206459E-02,-1.217980,1.177813,-1.896111/
```

```
C          DATA XVAL /1.9685E+03,1.9717E+03,1.9778E+03,1.9825E+03,
C       &  1.9873E+03,1.9903E+03/
C          DATA YVAL /5.8651E-09,2.035E-06,2.0350E-06,6.4463E-09,
C       &  1.7630E-06,5.8651E-09/
C          DATA SLOPE /1.782245,0.00,-1.217980,1.177813,-1.896111/


     &

C       Evaluate which slope to use and calculate y-value
        CALL ACRO_YEAR(USERX,XVAL(1),XVAL(KMAX+1),YEAR)
        DO I=1,KMAX
           IF ((YEAR.GE.XVAL(I)).AND.(YEAR.LT.XVAL(I+1)))
     &        ACRO_TIMELINE=EXP(ALOG(YVAL(I+1))-
     &                       (SLOPE(I)*(XVAL(I+1)-YEAR)))
        ENDDO

        RETURN
        END

        SUBROUTINE ACRO_YEAR(USERX,LOWERX,UPPERX,YEAR)

C Evaluates a given year to see if it falls within the range of 1967.9-1990.5.
C If it doesn't, USERX  is updated by either adding or subtracting a factor
C of 21.8 so that it does fall within the specified range.

C Declarations
        IMPLICIT NONE
        REAL USERX,LOWERX,UPPERX,DIFF,RMDR,YEAR
        INTEGER FACTOR

C Evaluate and modify USERX
        IF (USERX .LT. LOWERX) THEN
           DIFF=LOWERX-USERX
           RMDR=DIFF/21.8
           FACTOR=INT(RMDR)+1
           YEAR=USERX+REAL(FACTOR)*21.8

        ELSEIF (USERX .GT. UPPERX) THEN
           DIFF=USERX-UPPERX
           RMDR=DIFF/21.8
           FACTOR=INT(RMDR)+1
           YEAR=USERX-REAL(FACTOR)*21.8

        ELSE
           YEAR=USERX
        ENDIF


        RETURN
        END
```

```fortran
      REAL*4 FUNCTION BENDEL1(A,E)
C
C     Function evaluates the value of the proton-upset cross-section
C     at energy E using the Bendel 1-parameter inputs
C
C      Inputs:  A:    Bendel parameter A
C               E:    proton energy (in MeV)

C      Output:  SEU cross-section in 10E-12 cm2/bit
C
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C     Last update:   29 March 1996
C
C-----------------------------------------------------------------------
C
      IMPLICIT NONE
      REAL*4 Y,A,E

      BENDEL1 = 0.

      Y = (SQRT(18./A))*(E-A)
      IF (Y.LT.0.) Y=0.
      BENDEL1 = ((24./A)**14.)*((1.-EXP(-.18*SQRT(Y)))**4.)
      IF (BENDEL1.LT.0.) BENDEL1=0.0

      RETURN
      END
```

```
      REAL*4 FUNCTION BENDEL2(A,B,E)
C
C     Function evaluates the value of the proton-upset cross-section
C     at energy E using the Bendel 2-parameter inputs
C
C      Inputs:  A, B:  Bendel parameters A and B
C               E:     proton energy (in MeV)

C      Output:  SEU cross-section in 10E-12 cm2
C
C     Written by:     Allan J. Tylka
C                     Code 7654
C                     Naval Research Laboratory
C                     Washington, DC 20375-5352
C                     tylka@crs2.nrl.navy.mil
C
C     Last update:    29 March 1996
C
C---------------------------------------------------------------------------
C
      IMPLICIT NONE
      REAL*4 Y,A,B,E

      BENDEL2=0

      Y = (SQRT(18./A))*(E-A)
      IF (Y.LT.0.) Y=0.
      BENDEL2 = ((B/A)**14.)*((1.-EXP(-.18*SQRT(Y)))**4.)
      IF (BENDEL2.LT.0.) BENDEL2=0.0

      RETURN
      END
```

```
C***********************************************************************
      subroutine blccoords(lat, lon, alt, year, imod, bob0, L, yearp, B)
C***********************************************************************
c      Inputs:
c          lat, latitude
c          lon, east longitude
c          alt, altitude in km
c          year, not currently used
c          imod, = 1 for solar min model, = 2 for solar max model (these are the
c                  only choices provided for this first version of blccoords)
c      Outputs:
c           bob0 returned as Tylka requested (see caveat below where calculated)
c           L
c           yearp, year used by allmag (1964 for solar min, 1970 for solar max)
c           B
C***********************************************************************
      implicit none
      save

      real*4 lat, lon, alt, year, bob0, L, yearp, b

      real*8 lat8, lon8, alt8, err8, L8, b8

      integer *4 imod, imodold, model

      real*8 constem
      common /gmagmo/ constem

      data err8/ .1/
      data imodold /-10/

      if (imodold .eq. -10) then   ! first time
         imodold = imod
         if (imod .eq. 1) then
            yearp = 1964
         elseif (imod .eq.2) then
            yearp = 1970
         else
             stop 'blccords0'
         endif
         call stmag(imod, yearp)
      else
         if (imod. ne. imodold) then
            !imod has changed
            stop 'blccords1'
         endif
      endif

      lat8 = dble(lat)
      lon8 = dble(lon)
      alt8 = dble(alt)
      call invara (imod, yearp, lat8, lon8, alt8, err8, b8, L8)
      b = sngl(b8)
      L = sngl(l8)

c     compute b over b0 for Tylka....this may not be the bob0 value computed and
c      used by the trapped proton models because of checks on data base limits.
c      As an alternative, one can return actual value used from call to
c      subroutine Trapped_protons
      bob0 =  (b*(L*L*L)/ sngl(constem) )
```

```
      return
      end
c***********************************************************************
      subroutine allmag(model, tm, rkm, st, ct, sph, cph, br, bt, bp, b)
c***********************************************************************
      implicit none
      save

      real*8 ar, aor, b, br, bt, bp, const, cp, cph, ct, dp
      real*8 fn, fm, g, p, par, rkm, sp, sph, st, temp

      real * 4 tm

      integer*4 nmax, m, model, n

      common/magcof/ g(14,14), fn(14), fm(14), const(14,14), nmax
      dimension p(14,14), dp(14,14), sp(14), cp(14)
      data p(1,1), cp(1), dp(1,1), sp(1) / 2*1.,2*0. /

      sp(2) = sph
      cp(2) = cph
      do m = 3,nmax
         sp(m) = sp(2) * cp(m-1) + cp(2) * sp(m-1)
         cp(m) = cp(2) * cp(m-1) - sp(2) * sp(m-1)
      enddo
      aor = 6371.2/rkm
      ar = aor * aor * aor
      p(2,1) = ct
      dp(2,1) = -st
      p(2,2) = st
      dp(2,2) = ct
      br = -(ar+ar)*(g(2,1)*p(2,1)+p(2,2)*(g(2,2)*cp(2)+g(1,2)*sp(2)))
      bt = ar * (g(2,1)*dp(2,1)+dp(2,2)*(g(2,2)*cp(2)+g(1,2)*sp(2)))
      bp = ar * (g(1,2) * cp(2) - g(2,2) * sp(2)) * p(2,2)

      do n = 3, nmax
         ar = aor*ar
         do  m = 1,n
            if(m.ne.n) then
               p(n,m) = ct * p(n-1,m) - const(n,m) * p(n-2,m)
               dp(n,m) = ct * dp(n-1,m)-st*p(n-1,m)-const(n,m)*dp(n-2,m)
             else
               p(n,n) = st * p(n-1,n-1)
               dp(n,n) = st * dp(n-1,n-1) + ct * p(n-1,n-1)
             endif
            par = p(n,m) * ar
            if(m.ne.1) then
               temp = g(n,m) * cp(m) + g(m-1,n) * sp(m)
               bp = bp - (g(n,m)*sp(m)-g(m-1,n)*cp(m)) * fm(m) * par
             else
               temp = g(n,m)
            endif
            br = br - temp * fn(n) * par
            bt = bt + temp * dp(n,m) * ar

         enddo
      enddo
      bp = bp/st/100000.
      br = br/100000.
      bt = bt/100000.
```

```
      IF (abs(bp) .LT. 1.0E18 .AND. abs(br) .LT. 1.0E18 .AND.
     &    abs(bt) .LT. 1.0E18) THEN
        b =  sqrt(br*br + bt*bt + bp*bp )
      ELSE
        b =  1.0E18
      ENDIF

      return
      end
c*********************************************************************
      subroutine carmla (B, xi, vl)
c*********************************************************************
      implicit none
      save

      real*8 B, constem, gg, vl, xi, xx
c     compute l
C     *******************************************************
C     Equations Containing Constant Mag Moment Will Be Commented
C     Out And Rewritten With New Mag Moment Calculated In Stmag
C     Subroutine    3/15/91
C     *******************************************************
      common /gmagmo/ constem

      if( xi-1.0d-36 .le. 0.) then
        vl=(constem/B)**(1./3.)
        return
      endif

      xx = 3.0 * dlog(xi)
      xx = xx + dlog(B/constem)
      if (xx+22. .le. 0.) then
         gg = .333338*xx+.30062102
         go to 7
      endif
      if(xx+3.  .le. 0.) then
        gg=((((((((-8.1537735d-14*xx+8.3232531d-13)*xx+1.0066362d-9)*xx+
     1  8.1048663d-8)*xx+3.2916354d-6)*xx+8.2711096d-5)*xx
     2  +1.3714667d-3)*xx+.015017245)*xx+.43432642)*xx+.62337691
        go to 7
      endif

      if(xx-3. .le. 0.0) then
         gg=((((((((2.6047023d-10*xx+2.3028767d-9)*xx-2.1997983d-8)*xx-
     1   5.3977642d-7)*xx-3.3408822d-6)*xx+3.8379917d-5)*xx +
     2   1.1784234d-3)*xx+1.4492441d-2)*xx+.43352788)*xx+.6228644d0
         go to 7
      endif
      if(xx-11.7 .le. 0.) then
         gg=((((((((6.3271665d-10*xx-3.958306d-8)*xx+9.9766148d-07)*xx-
     1   1.2531932d-5)*xx+7.9451313d-5)*xx-3.2077032d-4)*xx +
     2   2.1680398d-3)*xx+1.2817956d-2)*xx+.43510529)*xx+.6222355d0
         go to 7
      endif
      if(xx-23. .le. 0) then
         gg=(((((2.8212095d-8*xx-3.8049276d-6)*xx+2.170224d-4)*xx -
     1   6.7310339d-3)*xx+.12038224)*xx-.18461796)*xx+2.0007187
      else
```

```
         gg=xx-3.0460681
      endif
    7 vl = (((1.0+dexp (gg))*constem)/B)**(1./3.)

c     end compute l
      return
      end
c*********************************************************************
      subroutine intega(arc, beg, bend, b, jep, eco, fi)
c*********************************************************************
c     IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
      implicit none
      save
      real*8  a, arc, arg1, asum, b, bb, beg, bend, c, dn, eco, fi
      real*8  t, tb,  te, x2, x3

      integer*4 i, kk,jep

c     dimension arc(200),beg(200),bend(200),b(200),eco(200)
      dimension arc(1),beg(1),bend(1),b(1),eco(1)

      kk = jep
      if (kk .gt. 4) go to 20
      if (kk .eq. 4) kk = kk-1
      a  =  b(kk-1)/b(2)
      x2 = b(kk)/b(2)
      x3 = b(kk+1)/b(2)
      asum = arc(kk) + arc(kk+1)
      dn = arc(kk)*arc(kk+1)*asum
      bb = (-a*arc(kk+1) * (arc(kk)+asum)+x2*asum**2-x3*arc(kk)**2) /dn
      c = (a*arc(kk+1) - x2 * asum + x3 * arc(kk))/dn
      fi =.157079632d+01 * (1.0-a+bb*bb/(4.0*c)) / dsqrt(dabs(c))
      return

   20 t = dsqrt(1.0d0-bend(2)/b(2))
      fi = (2.0*t-dlog((1.0+t)/(1.0-t)))/eco(2)
      if (b(2)-bend(kk) .gt. 0.) kk=kk+1
      t = dsqrt(dabs(1.0-beg(kk)/b(2)))
      fi = fi-(2.0*t-dlog((1.0+t)/(1.0-t)))/eco(kk)
      kk = kk - 1
   22 do i = 3, kk
         arg1 = 1. - bend(i)/b(2)
         if (arg1 .le. 0.) then
            te = 1.d-5
         else
            te = dsqrt(arg1)
         endif
         arg1 = 1. - beg(i)/b(2)
         if (arg1 .gt. 0.) then
            tb = dsqrt(arg1)
         else
            tb = 1.d-5
         endif
         if (dabs(eco(i))-2.d-5 .le. 0.) then
            fi = fi + ((te+tb)*(arc(i)+arc(i+1)))/4.
         else
            fi = fi+(2.*(te-tb)-dlog((1.+te)*(1.-tb)/((1.-te)*(1.+tb))))
     1      /eco(i)
         endif
       enddo
```

```
   30 return
      end
c*********************************************************************
       subroutine invara(model, tm, flat, flong, alt, err, bb, fn)
c*********************************************************************
C ****  Note,Error In L Is Typically Less Than 10.*Err*L (Percent)
C ****  Flat=Latitude In Degrees  , Flong=Longitude In Degrees
C ****  Alt=Altitude=Distance From Surface Of Earth In Kilometers
c     IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
      implicit none
      save
      real*4 Tm

      real*8 alt, arc, asum,  b, bb, bco, beg, bend,  blog, cco, dclt
      real*8 dco,  dn,  dx,  eco, err, fl, flint
      real*8 flat, flong, fn, r1, r2, r3, sa, sc,  v, vp, vn

      integer*4 i, j, jep, jup,  model
      dimension v(3,3), b(200), arc(200), vn(3), vp(3), beg(200),
     1  bend(200), blog(200), eco(200), r1(3), r2(3), r3(3)

      v(1,2) = alt/6371.2
      v(2,2) = (90.-flat)/57.2957795
      v(3,2) = flong/57.2957795
      arc(1) = 0.
      arc(2) = (1.0+v(1,2)) * sqrt(err) * 0.3
      dclt = 1.5708-0.2007 * dcos(v(3,2) + 1.239)
      if (v(2,2) .gt. dclt) arc(2) = -arc(2)
      call starta(r1, r2, r3, b, arc, v, model, tm)
      do i = 1, 3
         vp(i) = v(i,2)
         vn(i) = v(i,3)
      enddo
      call linesa(r1, r2, r3, b, arc, err, j, vp, vn, model, tm)
      if (j .ge. 200) then
          fl = -1.0
          go to 18
      endif
      jup = j
      do  j =1, jup
         arc(j) = dabs(arc(j))
         blog(j) = dlog(b(j))
      enddo
      jep = jup-1
      do j = 2, jep
         asum = arc(j) + arc(j+1)
         dx = blog(j-1) - blog(j)
         dn = asum * arc(j) * arc(j+1)
         bco = ((blog(j-1) - blog(j+1)) * arc(j)**2 -dx *asum**2)/dn
         cco = (dx * arc(j+1) - (blog(j) - blog(j+1)) * arc(j)) / dn
         sa = .75 * arc(j)
         sc = sa + .25 * asum
         dco = blog(j-1) - cco * sa * sc
         eco(j) = bco + cco * (sa + sc)
         beg(j) = dexp(dco+eco(j) * .5 * arc(j))
         bend(j) = dexp(dco+eco(j) * .5  * (asum+arc(j)))
      enddo
      beg(jup) = bend(jep)
      bend(jup) = b(jup)
      eco(jup) = (2.0/arc(jup)) * dlog(bend(jup)/beg(jup))
```

```
      call intega(arc, beg, bend, b, jep, eco, flint)
      call carmla (b(2), flint, fl)
   18 bb = b(2)
      fn = fl
      return
      end
c***********************************************************************
      subroutine linesa(r1, r2, r3, b, arc, err, j, vp, vn, model, tm)
c***********************************************************************
c     implicit real*8(a-h,o-z),integer(i-n)
      implicit none
      save
      real*4 tm

      real*8 a1, a2, a3, aa, aab, ad, am, ao6, arc, arcj, asum,  b
      real*8 bb, bd, bp, br, bt, cc, cd, cop, cot, cre, dd, dn, err
      real*8 pre1, pre2, pre3, qrt, r1, r2, r3
      real*8  ra, rbar, rkm,  rt, sip, sit, sna, ssq, vp, vn, x

      integer*4 i, ilp, is, j, model

c     dimension b(200), arc(200), r1(3), r2(3), r3(3), vn(3), vp(3), ra(3)
      dimension b(1), arc(1)  !subroutine arguments
      dimension r1(3), r2(3), r3(3), vn(3), vp(3), ra(3)
      cre = 0.25
      if (err .lt. 0.15625)  cre = (err**0.333333333)
      a3 = arc(3)
      aab = dabs(a3)
      sna = a3/aab
      a1 = arc(1)
      a2 = arc(2)
      ao6 = a3*a3/6.0
      j = 3
      ilp = 1
      is = 1
      GO TO 87

   66 is = 1
      j = j+1
      ao6 = a3 * a3/6.0
      arcj = a1 + a2 + a3
      ad = (asum+a1)/aa
      bd = asum/bb
      cd = a1/cc
   36 do i = 1, 3
         dd = r1(i)/aa-r2(i)/bb+r3(i)/cc
         if (is .eq. 1) then
           rt = r1(i) - (ad*r1(i)-bd*r2(i)+cd*r3(i)-dd*arcj) * arcj
            ra(i) = r1(i)
            r1(i) = r2(i)
            r2(i) = r3(i)
            r3(i) = rt
            vp(i) = vn(i)
         endif
        rbar = (r2(i) + r3(i))/2. - dd * ao6
         vn(i) = vp(i) + a3 * rbar
       enddo

   87 if (vn(2) .lt. 0.) vn(2) = -vn(2)
```

```
   77 if (vn(2) .le. 3.141592653) go to 78
      vn(2) = 6.283185307 - vn(2)
      go to 77
   78 if(vn(3) .ge. 0.) go to 81
      vn(3) = vn(3) + 6.283185307
      go to 78
   81 if(vn(3) .le. 6.283185307) go to 82
       vn(3) = vn(3) - 6.283185307
      go to 81
   82 go to (9, 10), is

    9 sit = dabs(dsin(vn(2)))
      pre1 = vn(1)
      pre2 = pre1 * vn(2)
      pre3 = pre1 * sit * vn(3)
  100 rkm = vn(1) * 6371.2

      IF (rkm .lt. 100.0) rkm=100.0

      ssq = sit * sit
      cot = dcos(vn(2))
      sip = dsin(vn(3))
      cop = dcos(vn(3))
      call allmag(model, tm, rkm, sit, cot, sip, cop, br, bt, bp, b(j))

C     Added error checking on b(j), 11-24-97.

      IF (b(j) .EQ. 0.0) b(j) = 1.0E-10  !avoid underflows, 11-24-97.
        r3(1) = br/b(j)
        dn = b(j) * vn(1)
        r3(2) = bt/dn
        r3(3) = bp/(dn*sit)
      asum = a3+a2
      aa = asum*a2
      bb = a3*a2
      cc = asum*a3
      is = 2
      go to 36

   10 sit = dabs(dsin(vn(2)))
      b(j) = b(j) * ((pre1/vn(1))**3)
      qrt = 0.5d0 * dabs(r3(1))/(0.1d0+dabs(r3(2)*vn(1)))
      x = (dabs(vn(1)-pre1)+qrt*dabs(vn(1)*vn(2)-pre2)+dabs(vn(1)*sit*
     1 vn(3)-pre3))/(aab*err*dsqrt(1.+qrt*qrt))
      go to (90, 93, 90), ilp
   93 if (x - 3.3) 90, 89, 89
   89 a3 = a3 * 0.2 * (8.0+x)/(0.8+x)
      j = j - 1
      ilp = 3
      asum = a2+a1
      aa = asum * a1
      bb = a2 * a1
      cc = asum * a2
      do i = 1, 3
         vn(i) = vp(i)
         r3(i) = r2(i)
         r2(i) = r1(i)
         r1(i) = ra(i)
      enddo
      go to 73
```

```
   90 if(j .gt. 200) go to 60
      a1 = a2
      if(b(j)-b(2) .gt. 0 ) go to 60
CF?      if(b(j)-b(2) .gt. 0 .OR. b(j) .EQ. 0.0) go to 60  !changed 11-24-97.
      ilp = 2
      a2 = a3
      a3 = a3 * .2 * (8.+x)/(.8+x)
      am = (2.-r3(2) * vn(1)) * vn(1) * cre
      if(dabs(a3)-am .gt. 0.) a3 = sna * am

      if(sna * r3(1)+.5 .gt. 0.) go to 73
       am = -.5 * sna * vn(1)/r3(1)
      if(dabs(a3)-am .gt. 0) a3 = sna * am
   73 arc(j+1) = a3
      aab = dabs(a3)
      go to 66

   60 return
      end
c*********************************************************************
      subroutine starta(r1, r2, r3, b, arc, v, model, tm)
c*********************************************************************
c     implicit real*8(a-h,o-z),integer(i-n)
      implicit none
      save
      real*8  aer, arc, b, bp, br, bt,  cop, cot,  dn
      real*8  oer, sip, sit, ssq, r1, r2, r3,  rkm, v

      real*4 tm

      integer*4 i, is,  model
c     dimension b(1), arc(1), v(3, 3), r1(3), r2(3), r3(3)
      dimension b(1), arc(1), v(3, 3), r1(1), r2(1), r3(1)   ! arguments

      sit = dabs(dsin(v(2,2)))
      aer = v(1,2)
      ssq = sit * sit
      oer = (6356.912+ssq * (21.3677+.108 * ssq))/6371.2
      v(1,2) = aer + oer
   10 if (v(3,2) .ge. 0. ) go to 12
      v(3,2) = v(3,2) + 6.283185307
      go to 10

   12 rkm = v(1,2) * 6371.2
c     if (model.eq.6) rkm = rkm+14.288-ssq * (21.3677+.108 * ssq)   !CHECK ORIG MODEL NO;'S
      cot = dcos(v(2,2))
      sip = dsin(v(3,2))
      cop = dcos(v(3,2))
      call allmag (model, tm, rkm, sit, cot, sip, cop, br, bt, bp, b(2))
      r2(1) = br/b(2)
      dn = b(2) * v(1,2)
      r2(2) = bt/dn
      r2(3) = bp/(dn * sit)
      is = 0
    1 do  i = 1, 3
         v(i,1) = v(i,2)-arc(2) * r2(i)
      enddo
      sit = dabs(dsin(v(2,1)))
    3 rkm = v(1,1) * 6371.2
```

```
      ssq = sit * sit
c     if(model.eq.6) rkm = rkm+14.288-ssq * (21.3677+.108 * ssq)
      cot = dcos(v(2,1))
      sip = dsin(v(3,1))
      cop = dcos(v(3,1))
      call allmag (model, tm, rkm, sit, cot, sip, cop, br, bt, bp, b(1))
      if ( b(1)-b(2) .ge. 0.) go to 5
      arc(2) = -arc(2)
      go to 1

    5 r1(1) = br/b(1)
      arc(3) = arc(2)
      dn = b(1) * v(1,1)
      r1(2) = bt/dn
      r1(3) = bp/(dn * sit)
      do i=1, 3
         v(i,1) = v(i,2)-arc(2) * (r1(i)+r2(i))/2.
      enddo
      sit = dabs(dsin(v(2,1)))
      is = is+1
c     go to (3, 7), is
      if (is .eq. 1) go to 3
c   7 do i = 1, 3
      do i = 1, 3
         v(i,3) = v(i,2)+arc(3) * ((1.5) * r2(i)-.5 * r1(i))
      enddo
      return
      end
c***********************************************************************
      subroutine  stmag(model, tm)
c***********************************************************************
C     Constant Mag Moment Replaced With Calculated Mag Moment,
C      Using  Geomagnetic Field Expansion Coeffiecients - 3/15/91

c    Inputs  model    choice of 2 models - see below
c            rkm      geocentric distance in kilometers
c            tm       time in years for desired field
c            st,ct    sin + cos of geocentric colatitude
c  *         sph,cph  sin + cos of east longitude
c   Outputs  br,bt,bp geocentric field components in gauss
c            b        field magnitude in gauss
c     IMPLICIT REAL*8(A-H,O-Z)
      implicit none
      save
      real*8 const, constem, em,  f1, f2, f3, fm, fn, g
      real*8 rad, t, t0

      real*4 tm, tmold

      integer*4 jj, k, l, m, modold, n, nmax, nmx, model

      common /gmagmo/ constem

      integer*4 g1(13,13),gt1(13,13),gtt1(13,13), g2(13,13), gt2(13,13),
     1 gtt2(13,13), lg(13,13,2), lgt(13,13,2), lgtt(13,13,2)

      real*4  gg(13,13,2), ggt(13,13,2), ggtt(13,13,2), shmit(13,13)

      equivalence (g1,gg,lg), (gt1,ggt,lgt), (gtt1,ggtt,lgtt),
     2      (g2,lg(1,1,2)), (gt2,lgt(1,1,2)), (gtt2,lgtt(1,1,2))
```

```
      character*32 label(2)

      dimension t0(2), nmx(2)

      common/magcof/ g(14,14), fn(14), fm(14), const(14,14), nmax

      data  label(1)/'IGRF 1965.0 80-TERM 10/68      '/
      data  label(2)/'HURWITZ US C+GS 168-TERM 1970  '/

      data t0/1965.d+00,1970.d+00/
      data nmx/9,13/

C  *****  G1,Gt1  Igrf 1965.0            80-Term       10/68  Epoch 1965
      data g1 / 1, -30339,-1654,1297,958,-223,47,71,10,4*0,5758,-2123,
     A 2994,-2036,805,357,60,-54,9,4*0,-2006,130,1567,1289,492,246,4,0,
     B -3,4*0,-403,242,-176,843,-392,-26,-229,12,-12,4*0,149,-280,8,-265
     C ,256,-161,3,-25,-4,4*0,16,125,-123,-107,77,-51,-4,-9,7,4*0,-14,
     D 106,68,-32,-10,-13,-112,13,-5,4*0,-57,-27,-8,9,23,-19,-17,-2,12,
     E 4*0,3,-13,5,-17,4,22,-3,-16,6,56*0/
      data gt1 / 10, 153,-244,2,-7,19,-1,-5,1,4*0,-23,87,3,-108,2,11,-3,
     F -3,4,4*0,-118,-167,-16,7,-30,29,11,-7,6,4*0,42,7,-77,-38,-1,6,19,
     G -5,5*0,-1,16,29,-42,-21,0,-4,3,5*0,23,17,-24,8,-3,13,-4,0,-1,4*0,
     H-9,-4,20,-11,1,9,-2,-2,3,4*0,-11,3,4,2,4,2,3,-6,-3,4*0,1,-2,-3,-2,
     I -3,-4,-3,-3,-5,56*0/
      data gtt1 /1,168*0/

C  *****  G2,GT2  Hurwitz Us Coast + Geodetic S.  168-Term   Epoch 1970.
      data g2/10,-302059,-17917,12899,9475,-2145,460,734,121,107,-39,16,
     A -4,57446,-20664,29971,-20708,8009,3595,651,-546,77,57,-26,-31,30,
     B -20582,430,16086,12760,4579,2490,95,46,-32,23,7,-36,5,-3699,2456,
     C -1880,8334,-3960,-290,-2188,175,-124,-110,-19,37,-3,1617,-2758,
     D 185,-2788,2436,-1669,20,-210,-44,131,-15,-3,-13,157,1420,-1310,
     E -911,808,-582,-22,-32,45,33,74,-6,4,-171,1146,625,-323,-78,38,
     F -1125,143,34,2,46,-8,-14,-666,-265,-34,81,209,-240,-186,41,125,
     G 15,6,1,-12,121,-160,22,-176,46,189,-46,-187,94,9,-8,2,-12,-174,
     H 163,14,-27,-32,80,137,-4,-14,-4,22,-24,-1,27,19,0,35,-45,22,-31,
     I 56,-1,-63,14,4,10,-2,26,-26,-9,21,-1,18,-14,-28,-17,-14,6,-4,-3,
     J 4,9,-1,-10,26,-32,13,-6,-19,7,19,12/
      data gt2/10,231,-244,-19,-7,12,-7,0,3,4*0,-46,112,-1,-90,-6,7,6,
     K -3,3,4*0,-104,-166,40,-20,-36,12,14,3,4,4*0,72,21,-52,-54,-11,0,
     L 17,6,1,4*0,22,-5,14,-24,-23,-15,6,3,-1,4*0,1,25,-14,9,1,11,-3,2,
     M -3,4*0,-5,11,2,-3,7,22,-5,1,9,4*0,-17,-3,7,1,-2,-3,-2,-1,-2,4*0,
     N 2,-6,-3,-4,1,-2,-2,-1,6,56*0/
      data gtt2 /1,168*0/

      data shmit(1,1) / 0.0 /, tmold / -100./
      parameter (rad = 57.29577636718750)

C  *****  Begin Program

      if(model .lt. 1 .or. model .gt. 2) stop 'stmag1'

      if(shmit(1,1).eq.-1.)   go to 8   ! already initialized

      do  n = 1, 14
         fn(n) = n
         do  m = 1, 14
            fm(m) = m-1
            const(n,m) = float((n-2)**2-(m-1)**2) / ((2*n-3)*(2*n-5))
```

```
         enddo
      enddo
C  *****  Initialize * Once Only, First Time Subroutine Is Called
      shmit(1,1) = -1.
      do n = 2, 13
         shmit(n,1) = (2 * n-3) * shmit(n-1,1) / (n-1)
         jj=2
         do m = 2, n
            shmit(n,m) = shmit(n,m-1) *sqrt(float((n-m+1) * jj)/(n+m-2))
            shmit(m-1,n) = shmit(n,m)
            jj = 1
         enddo
      enddo

      do k = 1, 2
         f1 = lg(1,1,k)
         f2 = lgt(1,1,k)
         f3 = lgtt(1,1,k)
         nmax = nmx(k)
         l = 0
         do n = 1, nmax
            do m = 1, nmax
               gg(n,m,k) = lg(n,m,k) * shmit(n,m)/f1
               ggt(n,m,k) = lgt(n,m,k) * shmit(n,m)/f2
               ggtt(n,m,k) = lgtt(n,m,k) * shmit(n,m)/f3
            enddo  !m
         enddo  !n
      enddo  !k

    8 if (model.eq.modold)  return
C  *****  NOTE WRITE STATEMENT - NEW MODEL OR NEW TIME
      type 9,   model, label(  model) ,tm
    9 format(' model used is number',i2,2x,a32,'  for tm =',f9.3/)

      modold = model
      tmold = tm
      nmax = nmx(model)
      t = tm-t0(model)
      do n = 1, nmax
         do m = 1, nmax
           g(n,m) = gg(n,m,model)+t * (ggt(n,m,model)+ggtt(n,m,model)*t)
         enddo
      enddo
      em = sqrt(g(1,2)**2 + g(2,2)**2 + g(2,1)**2)
      constem = em/100000.0
      return
      end
```

```
      SUBROUTINE CALC_SEU_RATE(NBITS,SEU_RATE,DAY_RATE,
     &                          PERSECOND,PERDAY)
      IMPLICIT NONE
      REAL*4 NBITS,SEU_RATE,DAY_RATE,PERSECOND,PERDAY
C
C     INPUTS:
C               NBITS    = number of bits per device
C               SEU_RATE = calculated SEU rate (in upsets/bit/second)
C     OUTPUTS:
C               PERSECOND = upsets/device/second
C               PERDAY    = upsets/device/day
C               DAY_RATE  = upsets/bit/day

      PERSECOND=NBITS*SEU_RATE
      PERDAY=PERSECOND*24.0*3600.
      DAY_RATE=SEU_RATE*24.0*3600.

      RETURN
      END
```

```
      SUBROUTINE CAPITALIZE_STRING(STRING,ILONG)
C
C     Re-writes an input character string STRING of length ILONG
C     into all capitals.
C
      IMPLICIT NONE
      CHARACTER STRING
      INTEGER*4 ILONG,I
      IF (ILONG.GE.1) THEN
          DO 100 I=1,ILONG
          IF (ICHAR(string(i:i)) .GE. 97) THEN
           string(i:i) = CHAR(ICHAR(string(i:i)) - 32)
          ENDIF
 100      CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE CHECK_CREME96_VERSION(FILENAME,IVER)
C
C     Examines first line of input file, to get version number
C
      IMPLICIT NONE
      CHARACTER*80 FILENAME,ILINE
      CHARACTER*3 VERSIONLABEL
      INTEGER*4 NCHAR,IVER,STAT,CREME96_OPEN
      INTEGER*4 FILENO
      DATA FILENO/4/

C     Modified 7/29/96: Version 1.01
c      OPEN(UNIT=FILENO,FILE='USER:'//FILENAME,STATUS='OLD',
c     &     READONLY,SHARED)
      stat = creme96_open(filename,'user',fileno,'old')
      READ(FILENO,1) ILINE
 1    FORMAT(A80)

      NCHAR=3
      VERSIONLABEL=ILINE(76:78)

      IF (VERSIONLABEL.EQ.'   ') THEN
          IVER=0
      ELSE
          read(versionlabel,'(i3)') iver
      ENDIF

      CLOSE(FILENO)
      RETURN
      END
```

```
      SUBROUTINE CHECK_FILE(IFILETYPE,FILENAME,IACCEPT)
C
C     Subroutine for checking existence and acceptability of specified
C     input file.
C
      IMPLICIT NONE
      INTEGER*4 MHMAX,NLINES,MAXFILETYPE
      PARAMETER (MAXFILETYPE=8)
      CHARACTER*80 ILINES,TEMPLINE
      PARAMETER (MHMAX=20)
      DIMENSION ILINES(MHMAX)
      INTEGER*4 IFILETYPE,JFILETYPE,J,IACCEPT,IANSWER,NHMAX,IHMAX
      DIMENSION IHMAX(MAXFILETYPE)
      DATA IHMAX/3,3,10,15,20,20,4,1/
      CHARACTER*80 FILENAME
      LOGICAL IEXIST,FILE_CHECKS,CREME96_INQUIRE

      INTEGER*4 IERR
      DATA IERR/0/
C
C
      IACCEPT=0
      CALL GET_CHECK_CONTROL(FILE_CHECKS)
      IF (.not. FILE_CHECKS) RETURN

C     First, see if specified input file exists:
c      INQUIRE(FILE='USER:'//FILENAME,EXIST=IEXIST)
      iexist = creme96_inquire(filename,'user')
      IF (.NOT. IEXIST) THEN
          WRITE(6,999)
  999     FORMAT(1X,' This file was not found in USER area.',
     &    '  Please try again.')
          CALL SHOW_DIRECTORY(IFILETYPE)
          IACCEPT=-1
          RETURN
      ENDIF
C
      IF (IFILETYPE.EQ.0) RETURN

C     Now see if file has correct type:
      CALL CHECK_FILE_TYPE(FILENAME,JFILETYPE)
      IF (JFILETYPE.GT.0) THEN
          IF (IFILETYPE.EQ.1) THEN
              IF (JFILETYPE.NE.1) THEN
                WRITE(6,9001)
 9001           FORMAT(1x,' This file does not contain a trapped',
     &          ' proton flux.  Please try again.')
                CALL SHOW_DIRECTORY(IFILETYPE)
                IACCEPT=-1
               ENDIF
          ELSEIF (IFILETYPE.EQ.2) THEN
              IF (JFILETYPE.NE.2) THEN
                WRITE(6,9002)
 9002         FORMAT(1x,' This file does not contain a geomagnetic',
     &        ' transmission function.',/,1x,' Please try again:')
                CALL SHOW_DIRECTORY(IFILETYPE)
                IACCEPT=-1
               ENDIF
          ELSEIF (IFILETYPE.EQ.3 .or. IFILETYPE.EQ.4) THEN
               IF (JFILETYPE.NE.3 .and. JFILETYPE.NE.4
```

```
     &              .and. JFILETYPE.NE.1) THEN
                  WRITE(6,9003)
9003              FORMAT(1x,' This file does not contain particle',
     &            ' fluxes.  Please try again.')
                  CALL SHOW_DIRECTORY(IFILETYPE)
                  IACCEPT=-1
                 ENDIF
           ELSEIF (IFILETYPE.EQ.5) THEN
                IF (JFILETYPE.NE.5) THEN
                  WRITE(6,9005)
9005              FORMAT(1x,' This file does not contain an integral',
     &            ' LET spectrum.  Please try again.')
                  CALL SHOW_DIRECTORY(IFILETYPE)
                  IACCEPT=-1
                ENDIF
           ELSEIF (IFILETYPE.EQ.6) THEN
                IF (JFILETYPE.NE.6) THEN
                  WRITE(6,9006)
9006              FORMAT(1x,' This file does not contain a differential',
     &            ' LET spectrum.  Please try again.')
                  CALL SHOW_DIRECTORY(IFILETYPE)
                  IACCEPT=-1
                ENDIF
          ELSEIF (IFILETYPE.EQ.7) THEN
                IF (JFILETYPE.NE.7) THEN
                  WRITE(6,9007)
9007              FORMAT(1x,' This file does not contain a shielding',
     &            ' distribution prepared by the CREME96 software.',
     &          /,' Please try again.')
                  CALL SHOW_DIRECTORY(IFILETYPE)
                  IACCEPT=-1
                ENDIF
          ELSEIF (IFILETYPE.EQ.8) THEN
                IF (JFILETYPE.NE.0 .and. JFILETYPE.NE.8) THEN
                  WRITE(6,9008)
9008              FORMAT(1x,' This file does not contain a',
     &            ' cross-section table.',
     &          /,' Please try again.')
                  CALL SHOW_DIRECTORY(IFILETYPE)
                  IACCEPT=-1
                ENDIF
          ENDIF
       IF (IACCEPT.LE.-1) RETURN
       ENDIF

       IF (IFILETYPE.GT.0 .and. IFILETYPE.LE.MAXFILETYPE) THEN
       NHMAX=IHMAX(IFILETYPE)
       CALL UNLOAD_HEADERS(FILENAME,NHMAX,ILINES,NLINES)
       DO 100 J=1,NLINES
          TEMPLINE=ILINES(J)
          IF (TEMPLINE(2:2).EQ.'%') WRITE(6,997) TEMPLINE
997       FORMAT(A80)
100    CONTINUE
       ENDIF

101    CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,996)
996    FORMAT(1x,' Is this the input file you want here? 0=no,1=yes')
       READ(*,*,ERR=101,IOSTAT=IERR) IANSWER
```

```
      IF (IANSWER.NE.1) THEN
         WRITE(6,995)
995      FORMAT(1x,' Please try again.')
         CALL SHOW_DIRECTORY(IFILETYPE)
         IACCEPT=-1
      ENDIF

      RETURN
      END
```

```
      PROGRAM CHECK_FILE_DRIVER
C
C     Stand-alone version of the CHECK_FILE routine, primarily written
C     for interface to the WWW version of CREME96.
C
C     In response to questions, USER supplies the following information
C        IFILETYPE: indicates desired type of file (1=.trp, etc.)
C        FILENAME:  filename (without directory name appended).
C
C     Outputs from the program which are to be subsequently displayed by
C     the WWW interface are prefaced with preface "**")
C
      IMPLICIT NONE
      INTEGER*4 IFILETYPE,JFILETYPE
      CHARACTER*80 FILENAME
      LOGICAL IEXIST,CREME96_INQUIRE
      INTEGER*4 IERR,K
      CHARACTER*30 DESCRIP
      DIMENSION DESCRIP(11)
      DATA DESCRIP/' a trapped proton flux',
     &             ' a geomag transmission fcn',
     &             ' external particle fluxes',
     &             ' internal particle fluxes',
     &             ' an integral LET spectrum',
     &             ' a differential LET spectrum',
     &             ' a shielding distribution',
     &             ' a cross-section table',
     &             ' a PUP-SEU report',
     &             ' a HUP-SEU report',
     &             ' a dose report'/
      INTEGER*4 NHMAX,LINEMAX
      PARAMETER (NHMAX=30)
      CHARACTER*80 HEADER_LINE
      DIMENSION HEADER_LINE(NHMAX)
      CHARACTER*80 ILINE,ILINEOUT
C
C-----------------------------------------------------------------------
C
C     Get inputs from user:
 101  CONTINUE
      WRITE(6,1000)
 1000 FORMAT(' Specified desired filetype: ',
     &     /,'   1=.TR*; 2=.GT*; 3=.FLX; 4=.TFX; 5=.LET; 6=.DLT;',
     &         ' 7=.SHD; 8=.XSD;',
     &     /,'   9=.PUP; 10=.HUP, 11=.DSE')
      READ(*,*,ERR=101,IOSTAT=IERR) IFILETYPE
 102  CONTINUE
      IF (IFILETYPE.LT.1 .or. IFILETYPE.GT.11) THEN
         WRITE(6,1005) IFILETYPE
 1005    FORMAT(1x,'ERROR: IFILETYPE = ',I8,' not defined. Try again.')
         STOP
      ENDIF
      WRITE(6,1100)
 1100 FORMAT(' Enter FILENAME: ')
      READ(*,105,ERR=102,IOSTAT=IERR) FILENAME
 105  FORMAT(A80)

C-----------------------------------------------------------------------
C
C     Now begin analysis of file:
```

```
C      First, see if specified input file exists:
c       INQUIRE(FILE='USER:'//FILENAME,EXIST=IEXIST)
       iexist = creme96_inquire(filename,'user')
       IF (.NOT. IEXIST) THEN
           WRITE(6,9101) FILENAME(1:78)
 9101      FORMAT(1x,'**',A78,
     %          /,1x,'**not found in USER area.  Try again.')
           STOP
       ELSE
           WRITE(6,9102) FILENAME(1:78)
 9102      FORMAT(1x,'**',A78)
       ENDIF
C
C      Now see if filetype matches requested type.

       IF (IFILETYPE.EQ.8) THEN
           WRITE(6,9105)
 9105      FORMAT(1x,'**According to the extension (.xsd), this file',
     %          /,1x,'**contains a user-supplied cross-section table.',
     %          /,1x,'**Here are the first 2 lines of the file: ')

       ELSEIF (IFILETYPE.NE.8) THEN

           CALL CHECK_FILE_TYPE(FILENAME,JFILETYPE)

           IF (IFILETYPE.EQ.JFILETYPE) THEN
               WRITE(6,9200) DESCRIP(IFILETYPE)
 9200          FORMAT(1x,'**This file contains',A30,'.')
           ELSEIF (IFILETYPE.NE.JFILETYPE) THEN
               WRITE(6,9201) DESCRIP(IFILETYPE)
 9201      FORMAT(1x,'**This file does NOT contain',A30,'. Try again:')
       ENDIF

       ENDIF


C      Now get header information:
       LINEMAX=0
       CALL UNLOAD_HEADERS(FILENAME,NHMAX,HEADER_LINE,LINEMAX)
       IF (LINEMAX.LE.0) THEN
               WRITE(6,9301)
 9301          FORMAT(1x,'**No header information stored in file.')
       ELSE
               IF (IFILETYPE.NE.8) WRITE(6,9302)
 9302          FORMAT(1x,'**Header information stored in this file:')
               DO 9400 K=1,LINEMAX
                  ILINE=HEADER_LINE(K)
                  ILINEOUT='**'//ILINE(3:80)
                  WRITE(6,9399) ILINEOUT(1:80)
 9399             FORMAT(1x,A80)
 9400          CONTINUE
       ENDIF


       STOP
       END
```

```
      SUBROUTINE CHECK_FILE(IFILETYPE,FILENAME,IACCEPT)
C
C     Subroutine for checking existence and acceptability of specified
C     input file.
C
      IMPLICIT NONE
      INTEGER*4 MHMAX,NLINES,MAXFILETYPE
      PARAMETER (MAXFILETYPE=8)
      CHARACTER*80 ILINES,TEMPLINE
      PARAMETER (MHMAX=20)
      DIMENSION ILINES(MHMAX)
      INTEGER*4 IFILETYPE,JFILETYPE,J,IACCEPT,IANSWER,NHMAX,IHMAX
      DIMENSION IHMAX(MAXFILETYPE)
      DATA IHMAX/2,3,10,15,20,20,4,1/
      CHARACTER*80 FILENAME
      LOGICAL IEXIST,NO_CHECKS,CREME96_INQUIRE
      DATA NO_CHECKS/.FALSE./

      INTEGER*4 IERR
      DATA IERR/0/
C
C
      IACCEPT=0
      IF (NO_CHECKS) RETURN

      RETURN
      END
```

```
      SUBROUTINE CHECK_FILE_TYPE(FILENAME,IFILETYPE)
C
C     Examines first line of input file, to check file-type code.
C
      IMPLICIT NONE
      CHARACTER*80 FILENAME
      INTEGER*4 IVER,STAT,CREME96_OPEN
      INTEGER*4 IFILETYPE,FILENO
      DATA FILENO/4/

      CALL CHECK_CREME96_VERSION(FILENAME,IVER)

      IFILETYPE=0
      IF (IVER.EQ.0) RETURN

c      OPEN(UNIT=FILENO,FILE='USER:'//FILENAME,STATUS='OLD',
c     &      READONLY,SHARED)
      stat = creme96_open(filename,'user',fileno,'old')

      READ(FILENO,1) IFILETYPE
 1    FORMAT(78x,I2)
      CLOSE(FILENO)

      RETURN
      END
```

```
      SUBROUTINE CHECK_HEADER_LENGTH(FILENAME,NHEADER)
C
C     Examines first line of input file, to get header length.
C
      IMPLICIT NONE
      CHARACTER*80 FILENAME,ILINE
      INTEGER*4 IVER,NHEADER,STAT,CREME96_OPEN
      INTEGER*4 FILENO
      DATA FILENO/4/

      CALL CHECK_CREME96_VERSION(FILENAME,IVER)

      IF (IVER.EQ.0.or.IVER.EQ.101) NHEADER=2
      IF (IVER.GE.102) THEN
c      OPEN(UNIT=FILENO,FILE='USER:'//FILENAME,STATUS='OLD',
c     &     READONLY,SHARED)
      stat = creme96_open(filename,'user',fileno,'old')
      READ(FILENO,*) NHEADER
      ENDIF

      CLOSE(FILENO)
      RETURN
      END
```

```fortran
      SUBROUTINE CHECK_NAME_CONFLICT(INFILE,OUTFILE,IACCEPT)
C
C     Makes sure that input and output names are not identical.
C
      IMPLICIT NONE
      CHARACTER*80 INFILE,OUTFILE,TEMPIN,TEMPOUT
      INTEGER*4 IACCEPT,ILONG
      LOGICAL FILE_CHECK

      IACCEPT=0
      CALL GET_CHECK_CONTROL(FILE_CHECK)
      IF (.not. FILE_CHECK) RETURN

C     Dispose of version numbers

      ILONG=INDEX(INFILE,';')
      TEMPIN=INFILE
      IF (ILONG.NE.0) TEMPIN=INFILE(1:ILONG-1)

      TEMPOUT=OUTFILE
      ILONG=INDEX(OUTFILE,';')
      IF (ILONG.NE.0) TEMPOUT=OUTFILE(1:ILONG-1)

C     Convert to upper case:

      ILONG=LEN(TEMPIN)
      CALL CAPITALIZE_STRING(TEMPIN,ILONG)
      ILONG=LEN(TEMPOUT)
      CALL CAPITALIZE_STRING(TEMPOUT,ILONG)

C     Now see if identical:
      IF (TEMPIN.EQ.TEMPOUT) THEN
          IACCEPT=-1
          WRITE(6,666) TEMPIN,TEMPOUT
 666      FORMAT(1x,' INFILE  = ',A69,/,1x,' OUTFILE = ',A69,
     &          /,1x,' ERROR: Input and Output filenames are identical.',
     &          /,1x,' Please enter another output filename.')
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE CHECK_OUTPUT_FILE(FILENAME,IACCEPT)
C
C     Subroutine for checking existence and specified output file.
C
      IMPLICIT NONE
      CHARACTER*80 FILENAME
      INTEGER*4 IACCEPT,IREPEAT
      LOGICAL IEXIST,FILE_CHECK,CREME96_INQUIRE
C
      IACCEPT=0
      CALL GET_CHECK_CONTROL(FILE_CHECK)
      IF (.not. FILE_CHECK) RETURN
      IEXIST=.FALSE.

C     See if specified output file already exists:

c      INQUIRE(FILE='USER:'//FILENAME,EXIST=IEXIST)
      iexist = creme96_inquire(filename,'user')

      IF (IEXIST) THEN
          WRITE(6,999)
 999      FORMAT(1X,' A file with this name',
     &                ' already exists in your USER area.',
     &           /,1x,' Do you wish to create a new file with',
     &                ' the same name? (0=no,1=yes)')
          READ(*,*,ERR=101) IREPEAT

          IF (IREPEAT.NE.1) THEN
            IACCEPT=-1
101         CONTINUE
            WRITE(6,995)
995         FORMAT(1x,' Please give another name: ')
          ELSE
            IACCEPT=0
            WRITE(6,996)
996         FORMAT(1x,' A new file with this same name will',
     &                  ' be created.')
          ENDIF

      ENDIF

      RETURN
      END
```

```fortran
      SUBROUTINE CHECK_RPP_DIMENSIONS(XM0,YM0,ZM0,
     &                                IPARAM,PARAMS,XSECT_FILE,
     &                                XM,YM,ZM)

C
C        Routine for extracting lateral RPP dimension from cross-section
C        when the input XM,YM values are zero:
C
C        Inputs:
C
C                  XM,YM,ZM = bit dimensions (in microns)
C                  IPARAM      = 1,2,4, indicating cross-section model
C                                1 = Bendel 1-parameter
C                                2 = Bendel 2-parameter
C                                4 = Weibull
C                                5 = Critical charge (pc)
C                                0 = table
C                  PARAMS(4)       = array containing cross-section parameters
C                  XSECT_FILE = file containing cross-section table.
C
C        Outputs:  XM,YM,ZM = revised RPP dimension
C
C
C        Written by:    Allan J. Tylka
C                       Code 7654
C                       Naval Research Laboratory
C                       Washington, DC 20375-5352
C                       tylka@crs2.nrl.navy.mil
C
C-----------------------------------------------------------------------
C
        IMPLICIT NONE
        INTEGER*4 IPARAM,NDUM,NTRY
        REAL*4 XM0,YM0,ZM0,PARAMS,XM,YM,ZM,LETMAX,XSMAX,DELTA_XS
        CHARACTER*80 XSECT_FILE
        DIMENSION PARAMS(4),LETMAX(2),XSMAX(2)
C
        XM=XM0
        YM=YM0
        ZM=ZM0
        IF (XM.GT.1.0E-6 .and. YM.GT.1.0E-6) RETURN
C
C       User has specified XM,YM=0.  Must extract value from the
C       cross-section inputs:
C
        IF (IPARAM.NE.5) THEN
        NDUM=2
        NTRY=1
        LETMAX(1)=1.0E+5
        LETMAX(2)=1.0E+6
 10     CONTINUE
        CALL EVALUATE_SEU_CROSS_SECTION(LETMAX,NDUM,IPARAM,PARAMS,
     &                                  XSECT_FILE,XSMAX)

        DELTA_XS=ABS(XSMAX(1)-XSMAX(2))/XSMAX(2)

           IF (DELTA_XS.GT.0.01) THEN
              WRITE(6,9999)
              NTRY=NTRY+1
              LETMAX(1)=LETMAX(1)*10.0
```

```
            LETMAX(2)=LETMAX(2)*10.0
            IF (NTRY.LE.10) GOTO 10
         ENDIF

      ELSEIF (IPARAM.EQ.5) THEN

          XSMAX(2)=PARAMS(2)

      ENDIF

      XM=SQRT(XSMAX(2))
      YM=XM
      IF (XM.LT.1.0E-6 .and. YM.LT.1.0E-6) THEN
          WRITE(6,9998) XM0,YM0,XSMAX(2)
9998      FORMAT(' Error in HUP inputs: ',
     &         /,' Input lateral RPP dimensions = ',2E13.6,
     &         /,' Input limiting cross-section = ',E13.6,
     &         /,' SEU RATE = 0.0 !!!')
      ENDIF

9999     FORMAT(' ERROR in CHECK_RPP_DIMENSIONS: Plateau not found.')

      RETURN
      END
```

```fortran
      SUBROUTINE CHECK_SHIELD_DISTRIBUTION(NSHIELD,UPATH0,FRACSHLD0,
     &                                     UPATH,FRACSHLD,
     &                                     XMEAN,XRMS,TOTAL,ERRFLAG)

      IMPLICIT NONE
      INTEGER*4 MAXSHIELD
      PARAMETER (MAXSHIELD=500)
      INTEGER*4 NSHIELD,K,INDX(MAXSHIELD)
      REAL*4 UPATH0,FRACSHLD0,UPATH,FRACSHLD
      REAL*4 XMEAN,XRMS,TOTAL
      INTEGER*4 ERRFLAG
      DIMENSION UPATH0(1),FRACSHLD0(1),UPATH(1),FRACSHLD(1)

      IF (NSHIELD.GT.MAXSHIELD) THEN
          WRITE(6,995) NSHIELD,MAXSHIELD
 995      FORMAT('@ 07001 ABNORMAL TERMINATION: ',
     &          /,1x,' ERROR in CHECK_SHIELD_DISTRIBUTION: ',
     &          /,1x,' TOO MANY BINS: ',I8,' > ',I8,' max.',
     &          /,1x,' STOP.')
          STOP
      ENDIF

C     First, check normalization
      TOTAL=0
      DO 100 K=1,NSHIELD
      TOTAL=TOTAL+FRACSHLD0(K)
 100  CONTINUE

      WRITE(6,999) NSHIELD
 999  FORMAT(' No. Shielding Bins  = ',I4)
      WRITE(6,998) TOTAL
 998  FORMAT(' Sum of shielding fractions = ',E13.6)
      ERRFLAG=0.0
      IF (ABS(TOTAL-1.0).GT.0.0001) THEN
            ERRFLAG=1
            WRITE(6,997)
 997        FORMAT(' Shielding distribution will be re-nomalized',
     &                ' to unit integral')
      ENDIF

      XMEAN=0.0
      XRMS=0.0
      DO 200 K=1,NSHIELD
C        Renormalize shielding fraction to unit integral:
         FRACSHLD0(K)=FRACSHLD0(K)/TOTAL
C        Calculate mean
         XMEAN=XMEAN+UPATH0(K)*FRACSHLD0(K)
C        Calculate mean square:
         XRMS= XRMS+FRACSHLD0(K)*UPATH0(K)**2
 200  CONTINUE
      XRMS=XRMS-XMEAN*XMEAN
      IF (XRMS.LT.0.0) XRMS=0.0
      XRMS=SQRT(XRMS)
      WRITE(6,250) XMEAN,XRMS
 250  FORMAT(1x,' Mean shielding thickness = ',E13.6,
     &     /,1x,' RMS deviation            = ',E13.6)

C     Now re-order according to increasing shielding thickness.  This
C     ordering makes the transport code more efficient.
```

```
      CALL INDEXX(NSHIELD,MAXSHIELD,UPATH0,INDX)

      DO 500 K=1,NSHIELD
              UPATH(K)      =UPATH0(INDX(K))
              FRACSHLD(K)=FRACSHLD0(INDX(K))
500   CONTINUE

      RETURN
      END
```

```
      SUBROUTINE COPY_HEADERS(INFILE,NHEADER,OUTUNIT)
C
C     Reads NHEADER lines of header information from file INFILE
C     and copies to unit OUTUNIT (which has previously been opened
C     by the calling routine).

      CHARACTER*80 INFILE,ILINE
      INTEGER*4 NHEADER,INUNIT,OUTUNIT,IVER,STAT,CREME96_OPEN
      DATA INUNIT/4/

      IF (NHEADER.LE.0) RETURN

      CALL CHECK_CREME96_VERSION(INFILE,IVER)

c      OPEN(UNIT=INUNIT,FILE='USER:'//INFILE,
c     &     STATUS='OLD',READONLY,SHARED)
      stat = creme96_open(infile,'user',inunit,'old')

      IF (IVER.LT.102) THEN

      DO J=1,NHEADER
         READ(INUNIT,999) ILINE
         WRITE(OUTUNIT,999) ILINE
 999     FORMAT(A80)
      ENDDO

      ELSEIF (IVER.GE.102) THEN
      READ(INUNIT,999) ILINE
      DO J=1,NHEADER
         READ(INUNIT,999) ILINE
         WRITE(OUTUNIT,999) ILINE
      ENDDO

      ENDIF

      CLOSE(INUNIT)
      RETURN
      END
```

```
      SUBROUTINE CREME96_DOSE(L,LETMIN,LETMAX,LETFLUX,MODEL_TYPE,
     &                        VERSION_NUMBER,PROGRAM_CODE,
     &                        DOSE_PER_SECOND,
     &                        ACCUMULATED_DOSE)
C
C     AJT 12/1/97:
C     Inputs:  L = number of bins in integral LET spectrum
C              LETMIN, LETMAX = lower, upper boundaries of LET (in MeV-cm2/g)
C              LETFLUX = integral particle flux (m2-s-sr)**-1
C              MODEL_TYPE = I*4 label of CREME96 environment type.
C     Outputs: VERSION_NUMBER:  CREME96 Version Number
C              PROGRAM_CODE:    DOSE program code
C              DOSE_PER_SECOND: Dose rate (rads/second)
C              ACCUMULATED_DOSE: krad/sec or krad, depending upon model type


      IMPLICIT NONE
      INTEGER*4 L,MAXSPEC
      REAL*4 LETMIN,LETMAX,LETFLUX,LETVAL
      DIMENSION LETFLUX(1)
      PARAMETER (MAXSPEC=5000)
      DIMENSION LETVAL(MAXSPEC)
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE,I,MODEL_TYPE
      REAL*4 DOSE_PER_SECOND,ACCUMULATED_DOSE
      REAL*4 DRAD,STP,DELTA,FOURPI
C
      WRITE(6,9998)
 9998 FORMAT(1x,' DOSE_DRIVER calculation started.',
     &          '  Please stand by.')

      DOSE_PER_SECOND=0.0
      ACCUMULATED_DOSE=0.0

      CALL GET_CREME96_VERSION(VERSION_NUMBER)
      PROGRAM_CODE=11
C
C     First evaluate corresponding LET Values
      IF (L.GT.MAXSPEC) THEN
          WRITE(6,995) L,MAXSPEC
 995      FORMAT('@ 11001 ABNORMAL TERMINATION: ',
     &          /,1x,' ERROR in CREME96_DOSE: ',
     &          /,1x,' TOO MANY BINS: ',I8,' > ',I8,' max.',
     &          /,1x,' STOP.')
          STOP
      ENDIF

      LETVAL(1)=LETMIN
      LETVAL(L)=LETMAX
      DO 100 I=2,L-1
         LETVAL(I)=LETMIN*(LETMAX/LETMIN)**(FLOAT(I-1)/FLOAT(L-1))
 100  CONTINUE

C     Now do dose calculation
      DRAD=0.0
      DO 200 I=2,L
        STP=SQRT(LETVAL(I)*LETVAL(I-1))
        DELTA=STP*(LETFLUX(I-1)-LETFLUX(I))
        DRAD=DRAD+DELTA
 200  CONTINUE
```

```
C     Convert units:
C     (/m2-s-sr)*(MeV-cm2/g)   to  rad/sec
C     1 rad = 6.24E7 MeV/g

      FOURPI=16.0*ATAN(1.0)
      DOSE_PER_SECOND=DRAD*FOURPI/6.24E11

C     Now calculate ACCUMULATED DOSE:
C     For solar-quiet models, convert to rad/year
      IF (MODEL_TYPE.EQ.0) THEN
C         Calculate annual dose rate (krad/yr):
          ACCUMULATED_DOSE=DOSE_PER_SECOND*31557.6
      ELSEIF (MODEL_TYPE.EQ.2) THEN
C         Worst-week accumulated dose (krad; in 180 hours):
          ACCUMULATED_DOSE=DOSE_PER_SECOND*648.
      ELSEIF (MODEL_TYPE.EQ.1) THEN
C         Worst-day accumulated dose (krad; in 18 hours):
          ACCUMULATED_DOSE=DOSE_PER_SECOND*64.8
      ELSEIF (MODEL_TYPE.EQ.3) THEN
C         Peak Solar Dose rate (krad/sec):
          ACCUMULATED_DOSE=DOSE_PER_SECOND/1000.
      ENDIF

      WRITE(6,9999)
 9999 FORMAT(1x,' DOSE_DRIVER calculation completed.  Thank you.')
      IF(MODEL_TYPE.EQ.0) WRITE(6,9000) DOSE_PER_SECOND,ACCUMULATED_DOSE
      IF(MODEL_TYPE.EQ.1) WRITE(6,9001) DOSE_PER_SECOND,ACCUMULATED_DOSE
      IF(MODEL_TYPE.EQ.2) WRITE(6,9002) DOSE_PER_SECOND,ACCUMULATED_DOSE
      IF(MODEL_TYPE.EQ.3) WRITE(6,9003) DOSE_PER_SECOND,ACCUMULATED_DOSE

 9000 FORMAT(' Average Dose = ',1PE13.6,' rad/sec = ',1PE13.6,
     &       ' krad/year')
 9001 FORMAT(' Worst-day average dose rate = ',1PE13.6,' rad/sec',
     &      /,' Event-Accumulated Dose = ',1PE13.6,
     &       ' krad in 18.0 hours.')
 9002 FORMAT(' Worst-week average dose rate = ',1PE13.6,' rad/sec',
     &      /,' Event-Accumulated Dose = ',1PE13.6,
     &       ' krad in 180.0 hours.')
 9003 FORMAT(' Peak SEP dose rate = ',1PE13.6,' rad/sec = ',
     &        1PE13.6,' krad/sec')

      RETURN
      END
```

```
      SUBROUTINE CREME96_FLUX(IZLO,IZHI,ELOWER,EUPPER,YEAR,IMODE,ITRANS,
     *                        GTRANSFILE,TRAPDFILE,
     *                        VERSION_NUMBER,PROGRAM_CODE,
     *                        M,E,FLX)
C
C     Routine for generating the CREME96 particle fluxes.
C     Modified 7/29/96 to output version number & program code
C
      IMPLICIT NONE
      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 E,FLX
      DIMENSION E(MARR),FLX(NELM,MARR)

      REAL*4 ENERGY,DE,GET_CREME96_FLUX
      INTEGER*4 IZLO,IZHI,IMODE,ITRANS,J,K,M
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE
      REAL*4 ELOWER,EUPPER,YEAR
      CHARACTER*80 GTRANSFILE,TRAPDFILE

      WRITE(6,9998)
 9998 FORMAT(1x,' FLUX_DRIVER calculation started.  Please stand by.')

      M=1002
      CALL GET_CREME96_VERSION(VERSION_NUMBER)
      PROGRAM_CODE=3

C     Compute energies on logaritmically-spaced grid

      DE=(EUPPER/ELOWER)**(1./(M-1.))
      E(1)=ELOWER
      DO J=2,M-1
        E(J)=E(J-1)*DE
      END DO
      E(M)=EUPPER

      IF (ITRANS.GT.0) CALL LOAD_GTF(GTRANSFILE)
      IF (ITRANS.EQ.2) CALL LOAD_TRAPPED_PROTONS(TRAPDFILE)
      IF (ITRANS.GT.0 .and. IMODE.GE.1) CALL LOAD_SEP_QSTATES


C     Compute fluxes for each element

      DO J=IZLO,IZHI
        DO K=1,M
          ENERGY=E(K)
          FLX(J,K)=GET_CREME96_FLUX(J,ENERGY,YEAR,IMODE,ITRANS)
        END DO
      END DO

      WRITE(6,9999)
 9999 FORMAT(1x,' FLUX_DRIVER calculation completed.  Thank you.',
     & /,1x,' All fluxes are in units of particles/m2-s-sr-MeV/nuc.',
     &      ' vs. energy in MeV/nuc.',
     & /,1x,' Recommended next step:',
     &      ' TRANS (RUN CREME96:TRANSPORT_DRIVER)')

      RETURN
      END
```

```
      Logical function creme96_inquire(filename,path)

c     FILENAME:       The non-fully specified name of the target file.
c
c     PATH:           Contains the VMS logical or DOS environment variable
c                     pointing to file location
c
c     Calling example:
c
c             STAT = creme96_inquire('input.dat','creme96')
c
c     A return value of .TRUE. indicates that the target file was
c     found in the specified directory.  .FALSE. otherwise.


      IMPLICIT NONE
      character*80    file,creme96_full_filename
      character*(*)   filename
      character*(*)   path
      logical         exist

      file = creme96_full_filename(filename,path)

c     write(*,*)'In Inquire... fullname: ',file

      inquire(file=file,exist=exist)

c     write(*,*)'Exist: ',exist

      creme96_inquire = exist
      return
      end
```

```fortran
      SUBROUTINE CREME96_LETSPEC(LETMINMG,LETMAXMG,TARGET,
     &                           ELOWER,EUPPER,M,IZLO,IZUP,
     &                           INPUT_FLUX,
     &                           VERSION_NUMBER,PROGRAM_CODE,IDIFSPEC,
     &                           LETMIN,LETMAX,L,SPECT,DIFSPEC)
C
      IMPLICIT NONE
      REAL*4 LETMINMG,LETMAXMG,ELOWER,EUPPER
      REAL*4 LETMIN,LETMAX
      INTEGER*4 M,IZLO,IZUP,L
      CHARACTER*12 TARGET
      INTEGER*4 MARR,NELM,LARR
      PARAMETER (MARR=5000,NELM=92,LARR=1002)
      REAL*4 INPUT_FLUX(NELM,MARR),SPECT(LARR),DIFSPEC(LARR)
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE,IDIFSPEC
C

      WRITE(6,9998)
 9998 FORMAT(1x,' LETSPEC_DRIVER calculation started.',
     &          '  Please stand by.')


      CALL GET_CREME96_VERSION(VERSION_NUMBER)
      PROGRAM_CODE=5
C
C
C     Prepare for ULET/LETSPEC calculation.
C
C     Change units of LET range from /mg to /g
      LETMIN=1000.0*LETMINMG
      LETMAX=1000.0*LETMAXMG
C
C     Specifiy number of points in integral LET spectrum
      L=LARR
C
C     Now calculate integral LET spectrum:

      CALL ULET96(LETMIN,LETMAX,TARGET,
     &            ELOWER,EUPPER,M,IZLO,IZUP,
     &            INPUT_FLUX,L,SPECT)
C
C
C
C     Now calculate differential LET spectrum
      IF (IDIFSPEC.EQ.1)
     &CALL MAKE_DIFLET_SPECTRUM(LETMIN,LETMAX,L,SPECT,DIFSPEC)

      WRITE(6,9999)
 9999 FORMAT(1x,' LETSPEC_DRIVER calculation completed.  Thank you.',
     & /,1x,' Integral flux is in units of particles/m2-s-sr vs. LET',
     &      ' in MeV-cm2/gram.',
     & /,1x,' Recommended next step: HUP',
     &      ' (RUN CREME96:HI_UPSET_DRIVER)')
      IF (IDIFSPEC.EQ.1) WRITE(6,9997)
 9997 FORMAT(1x,' Differential LET spectrum of flux '
     &      ' (in particles/m2-s-sr-(MeV-cm2/gram))',
     &   /,1x,' vs. LET (in MeV-cm2/gram) also calculated.')

      RETURN
      END
```

```
      SUBROUTINE CREME96_TRANSPORT(INPUT_FLUX,
     &                             ELOWER,EUPPER,M,IZLO,IZUP,
     &                             IPATH,UPATH0,TARGET,SHIELDFILE,
     &                             VERSION_NUMBER,PROGRAM_CODE,
     &                             OUTPUT_FLUX)
C
C************************************************************************
C   This subroutine transports an input particle environment through a
C   specified thickness and type of shielding.  It takes account both
C   ionization energy loss (dE/dx) as well as energy-dependent nuclear
C   fragmentation.  The output is the particle enviroment (differential
C   fluxes vs. energy) inside the spacecraft, that is, 'behind' the specified
C   shielding.  This routine includes many refinements over the old CREME
C   transport routine ("INSIDE").  Specifically:
C
C   CREME96_TRANSPORT keeps track of projectile fragments; the old CREME
C   code ignored them.  This routine also uses improved Silberberg, Tsao,
C   and Barghouty energy-dependent fragmentation cross-sections.  Both of
C   these improvements can be important for thick shielding.
C
C   At present CREME96_TRANSPORT only does aluminum shielding; future
C   versions will also offer transport through other shielding materials.
C
C   CREME96_TRANSPORT is based on the "UPROP" code, as originally developed
C   by John R. Letaw of Severn Communication Corp. under contract to
C   the Gamma & Cosmic Ray Astrophysics Branch of Naval Research Laboratory
C   in 1989.  Significant improvements and "bug-extermination" have been
C   provided by A.F. Barghouty of Roanoke College.
C
C************************************************************************
C
      IMPLICIT NONE
      CHARACTER*12 TARGET
      CHARACTER*80 SHIELDFILE
      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 INPUT_FLUX(NELM,MARR),OUTPUT_FLUX(NELM,MARR)
      REAL*4 TEMP_INPUT(NELM,MARR),TEMP_FLUX(NELM,MARR)
      REAL*4 ELOWER,EUPPER,UPATH0,PATH,PSTEP,PSTEPMIN,PSTEPMAX
      REAL*4 PATHOLD,DELTA_PATH,TEMP_PATH
      INTEGER*4 M,N,NSP,IZLO,IZUP,IPATH,IULABEL
      REAL*4 UPATH,UUPATH,FRACSHLD
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE
      INTEGER*4 MAXSHIELD,NSHIELD,K,IELM,IARR
      PARAMETER (MAXSHIELD=500)
      DIMENSION UPATH(MAXSHIELD),FRACSHLD(MAXSHIELD)
      CHARACTER*5 UNITS_LABEL
      DIMENSION UNITS_LABEL(4)
      DATA UNITS_LABEL/'g/cm2','mils ','cm   ','!!!!!'/
C
C
      WRITE(6,9998)
 9998 FORMAT(1x,' TRANSPORT_DRIVER calculation started.',
     &          ' Please stand by.')


      CALL GET_CREME96_VERSION(VERSION_NUMBER)
      PROGRAM_CODE=4
C
C     Now set parametes for transport calculation.
```

```
C     Use recommended default:
C     Use energy-dependent fragmentation cross-sections
      N=10
C     Use straight-ahead approximation; ignore energy spread of target fragments
C     (This takes a lot of time and generally has only very small effect.)
      NSP=0
C
C     Set maximum & minimum PSTEP sizes allowed in transport
      PSTEPMIN=0.20
      PSTEPMAX=0.20


      IF (UPATH0.GT.0.0) THEN
          NSHIELD=1
          UPATH(1)=UPATH0
          FRACSHLD(1)=1.00
      ELSE
          CALL UNLOAD_SHIELDFILE(SHIELDFILE,
     &                               IPATH,NSHIELD,UPATH,FRACSHLD)
      ENDIF
      IULABEL=IPATH+1
      IF (IULABEL.GT.4) IULABEL=4

      PATHOLD=0.0

C
      DO 1000 K=1,NSHIELD
      WRITE(6,999) K,UPATH(K),UNITS_LABEL(IULABEL),FRACSHLD(K)
  999 FORMAT(1x,' SHIELDING BIN ',I4,' THICKNESS = ',F10.4,1x,A5,
     &          '   FRACTION = ',F8.4)

      UUPATH=UPATH(K)

C     Get shielding thickness (PATH) in g/cm2 and transport step size:
      CALL UNLOAD_PATH(IPATH,UUPATH,TARGET,PATH,PSTEPMIN,PSTEPMAX,PSTEP)
C
C     Now perform transport:
      IF (NSHIELD.EQ.1) THEN
      CALL UPROP96(INPUT_FLUX,
     &              ELOWER,EUPPER,M,IZLO,IZUP,
     &              N,NSP,PATH,PSTEP,TARGET,
     &              OUTPUT_FLUX)
C
C
      ELSE
C
C     Modification 8-16-96 by AJT:
C     To speed up calculations through thick shielding distributions,
C     allow output of one step to be input to the next step.

      DELTA_PATH=PATH-PATHOLD
      IF (DELTA_PATH.LT.0.) DELTA_PATH=0.0
      DO 300 IELM=1,NELM
         DO 200 IARR=1,MARR
                IF (K.EQ.1) THEN
                TEMP_INPUT(IELM,IARR)=INPUT_FLUX(IELM,IARR)
                TEMP_PATH=PATH
                ELSEIF (K.GT.1 .and. DELTA_PATH.LT.PSTEP) THEN
                TEMP_INPUT(IELM,IARR)=INPUT_FLUX(IELM,IARR)
                TEMP_PATH=PATH
```

```
                 ELSEIF (K.GT.1 .and. DELTA_PATH.GE.PSTEP) THEN
                 TEMP_INPUT(IELM,IARR)=TEMP_FLUX(IELM,IARR)
                 TEMP_PATH=DELTA_PATH
                 ENDIF
  200     CONTINUE
  300 CONTINUE


       CALL UPROP96(TEMP_INPUT,
      &              ELOWER,EUPPER,M,IZLO,IZUP,
      &              N,NSP,TEMP_PATH,PSTEP,TARGET,
      &              TEMP_FLUX)
       PATHOLD=PATH


C      Now add to weighted sum:
       DO 500 IELM=1,NELM
          DO 400 IARR=1,MARR
             OUTPUT_FLUX(IELM,IARR)=OUTPUT_FLUX(IELM,IARR)+
      &                                TEMP_FLUX(IELM,IARR)*FRACSHLD(K)
  400     CONTINUE
  500 CONTINUE

       ENDIF

 1000 CONTINUE
C
       WRITE(6,9999)
 9999 FORMAT(1x,' TRANSPORT_DRIVER calculation completed.  Thank you.',
      & /,1x,' All fluxes are in units of particles/m2-s-sr-MeV/nuc',
      &      ' vs. energy in MeV/nuc.',
      & /,1x,' Recommended next step: ',
      & /,5x,' LETSPEC',
      &      ' (RUN CREME96:LETSPEC_DRIVER)'
      &      ' for heavy-ion induced SEUs;',
      & /,2x,' or PUP     (RUN CREME96:PROTON_UPSET_DRIVER)',
      &      ' for proton-induced SEUs.')

       RETURN
       END
```

```
      REAL FUNCTION CRF96(IZ,EN,YEAR,IMODE)
C
C     THIS ROUTINE RETURNS THE DIFFERENTIAL FLUX IN PARTICLES/((M**2)*
C     STER*SEC*MEV/U) AS IT IS FOUND IN THE INTERPLANETARY MEDIUM
C     NEAR EARTH and OUTSIDE the magnetosphere
C
C     IZ = ATOMIC NUMBER OF THE IONS.
C     E = ENERGY (IN MEV/U).
C     Y = THE YEAR:  1975.144=SOLAR MIN; 1980.598=SOLAR MAX.
C     M = Particle environment model
C         0 = non-solar particles only: GCR+ACR
C         1 = "Worst day" Solar Energetic Particle Environment
C         2 = "Worst week" Solar Energetic Particle Environment
C         3 = "Peak (worst 5-minutes) Solar Energetic Particle Environment
C
C
      IMPLICIT NONE
      INTEGER*4 IZ,IQ,IMODE,IDUM
      REAL*4 EN,YEAR,GCRF,GCR_FLUX,ACRF,ACR_FLUX,SEP_FLUX

      CRF96=0.0
      IF (IMODE.LT.0. .or. IMODE.GT.3) RETURN
      IF (EN.LT.0.) RETURN
      IF (IZ.LT.1 .or. IZ.GT.92) RETURN

      IF (IMODE.EQ.0) THEN

C         Get Galactic Cosmic Ray contribution
          GCRF=GCR_FLUX(IZ,EN,YEAR,IDUM)

C         Get Anomalous Cosmic Ray contribution
          ACRF=0.0
          DO 100 IQ=1,IZ
          ACRF=ACRF+ACR_FLUX(IZ,IQ,EN,YEAR)
 100      CONTINUE

          CRF96=GCRF+ACRF
      ELSEIF (IMODE.NE.0) THEN
          CRF96=SEP_FLUX(IZ,EN,IMODE)
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE CTABLE(ELOWER,EUPPER,N,NSP,IZLO,IZUP,TARGET)
C***********************************************************************
C SUBROUTINE CTABLE in Module UPROP.FOR
C
C Creates the auxiliary spallation cross section data file (CTABLE.DAT) if
C it does not already exist.  It also calculates energy losses associated
C with spallation reactions (Sept. 1993).
C
C Modified 06-05-96:   add NSP to arguments, to control PARTIALS
C Modified 11-17-97:   add IMPLICIT NONE and variable-type declarations.
C***********************************************************************
C Parameters
C
C NELM    Maximum atomic number of elements to be transported (<= 109)
C MCS     Maximum number of energies at which cross section data are
C         defined
C ELOWER  Lower energy bound of input and output spectra (>= 0.1 MeV)
C EUPPER  Upper energy bound of input and output spectra (<= 100000 MeV)
C N       Number of logarithmically equally-spaced energy bins at which
C         cross sections are evaluated (ABS(N) < MCS)
C NSP     =1 turns on nuclear dE/dx in PARTIALS; 0 otherwise
C IZLO    Least atomic number of elements transported (>= 1)
C IZUP    Greatest atomic number of elements transported (<= 109)
C TARGET  Name of the target shielding material (<= 12 bytes)
C
C Important variables
C
C ECS     Energy at each cross section grid point.
C CT      Temporary array for holding the spallation cross sections of
C         one element at one energy.
C ELOSS   Temporary array for holding the energy loss of one element
C         at one energy averaged over each fragment's isotopes.
C TOTAL   Abundance weighted elemental cross section.
C C       Cross section array for all elements at a single energy.
C ENLOSS  Energy loss array for all elements at a single energy averaged
C         each element's isotopes.
C
C Subprograms
C
C SUBROUTINE MFP(ENERGY,K,ALL,TARGET,PATH)
C   Returns the mean free path PATH in g/cm**2 at energy ENERGY for an
C   element with charge K and mass ALL in target material TARGET
C
C SUBROUTINE PARTIALS(ENERGY,K,ALL,TARGET,CT,ANORM)
C   Returns the partial spallation mean free paths CT in g/cm**2 at energy
C   ENERGY for an element with charge K and mass ALL in target material
C   TARGET.
C
C BLOCK DATA D01
C   Defines the atomic masses of elements in the range 1 <= Z <= 109 and
C   places them in the array AMASS
C
C Data Files
C
C PERIODIC.DAT
C   Contains a list of the isotopes of each element and their natural
C   abundance.
C
C CTABLE.DAT
C   Contains nuclear spallation cross section data for the transport
```

```
C   calculation.  Automatically created by this subroutine when needed.
C***********************************************************************
       IMPLICIT NONE
       INTEGER NELM, MCS
       PARAMETER (NELM=92,MCS=10)

       REAL*4 CABU(NELM,9),TOTAL(NELM)
       REAL*4 ECS(MCS),C(NELM,NELM),CT(NELM),NORM(NELM)
       REAL*4 CS(NELM,NELM),ENLOSS(NELM,NELM),ELOSS(NELM),ETOTL(NELM)
       INTEGER*4 CISO(NELM,9),STAT,CREME96_OPEN
       CHARACTER*12 TARGET,TARGET$

       REAL*4 AMASS
       COMMON/MASS/AMASS(109)
       REAL*4 ELOWER$,EUPPER$,ELOWER,EUPPER
       INTEGER*4 N$,IZLO$,IZUP$,NSP$,N,IZLO,IZUP,NSP

       INTEGER*4 I,J,K,L
       REAL*4 ENERGY, DENERGY,ALL,ANORM, PATH, REALNORM,FPRO,FALP

       DATA ELOWER$,EUPPER$,N$,IZLO$,IZUP$,NSP$/0.,0.,0,0,0,0/
       DATA TARGET$/' '/

C      FORMAT Statements

C Format statement modified 6-5-96 by AJT to accomodate NSP
100    FORMAT(1X,2(1PE10.4,2X),4(I5,2X),A12,2X,1PE10.4)
200    FORMAT((1X,6(1PE11.4,2X)))
300    FORMAT(6X,9(1X,I3,1X,F4.2))

C      Otherwise perform calculation of cross sections
C      Read in list of elements from PERIODIC.DAT

c       OPEN(UNIT=15,FILE='CREME96:PERIODIC.DAT',STATUS='OLD',
c      &      READONLY,SHARED)
       stat = creme96_open('periodic.dat','cr96tables',15,'old')
       DO J=1,83
         READ(15,300) (CISO(J,K),CABU(J,K),K=1,9)
       END DO
       CLOSE(UNIT=15)

C      Open output data file

c       OPEN(UNIT=13,FILE='USER:CTABLE.DAT',STATUS='NEW')
        stat = creme96_open('ctable.dat','user',13,'new')
c       OPEN(UNIT=17,FILE='USER:SPTABLE.DAT',STATUS='NEW')
        stat = creme96_open('sptable.dat','user',17,'new')

C      Write header

       ELOWER$=ELOWER
       EUPPER$=EUPPER
       N$=N
       IZLO$=IZLO
       IZUP$=IZUP
       NSP$=NSP
       TARGET$=TARGET
       WRITE (13,100) ELOWER,EUPPER,N,IZLO,IZUP,NSP,TARGET
       WRITE (13,'(A)') ' '
       WRITE (17,100) ELOWER,EUPPER,N,IZLO,IZUP,NSP,TARGET
```

```
      WRITE (17,'(A)') ' '

C     Compute vector of energies

      IF (N.GE.2) THEN
        ECS(1)=ELOWER
        DENERGY=(EUPPER/ELOWER)**(1./FLOAT(N-1))
        DO J=2,N
          ECS(J)=ECS(J-1)*DENERGY
        END DO
      ELSE
        ECS(1)=2000.
      ENDIF

C     Compute parameters

      DO J=1,N
        ENERGY=ECS(J)
C     Initialize some arrays

        DO K=1,NELM
          TOTAL(K)=0.
          NORM(K)=0.
          DO I=1,NELM
            C(K,I)=0.
            ENLOSS(K,I)=0.
            CS(K,I)=0.
          END DO
        END DO

        DO K=IZLO,IZUP

C     For each incident particle at each energy compute
C     partial cross sections, total cross section, and the
C     normalization factor.  Average over isotopes.

          DO L=1,9
            IF (CABU(K,L).GT.0.) THEN
            ALL=REAL(CISO(K,L))
            CALL PARTIALS(ENERGY,K,ALL,TARGET,CT,NSP,ELOSS,ANORM)
            CALL MFP(ENERGY,K,ALL,TARGET,PATH)
            DO I=2,K+1
              C(I,K)=C(I,K)+CT(I)*CABU(K,L)
              CS(I,K)=C(I,K)   !Cross section w/o merging!
            END DO
            DO I=1,K+1    !Energy-loss averaged over isotopes of K:
              ENLOSS(I,K)=ENLOSS(I,K)+ELOSS(I)*CABU(K,L)
            END DO
            NORM(K)=NORM(K)+ANORM*CABU(K,L)
            TOTAL(K)=TOTAL(K)+PATH*CABU(K,L)
            ENDIF
          END DO

C     Allow for further renormalization

      REALNORM=1.

C     Compute cross sections for proton and alpha production
C     The procedure used here is taken from J.R. Letaw,
C     Phys. Rev. C28, 2178 (1983).
```

```
          IF (K.GT.2) THEN
C           FF=AMASS(K)/FLOAT(K)+0.67
C           ASSUME 15% He AND 85% H in product
C           FPRO=1./(1.+2a/p)
            FPRO=0.739
C           FALP=1./(2.+ p/a)
            FALP=0.130
            C(1,K)=FPRO*(TOTAL(K)*FLOAT(K)-NORM(K)*REALNORM)
            C(2,K)=FALP*(TOTAL(K)*FLOAT(K)-NORM(K)*REALNORM)+C(2,K)
          ENDIF

C     Compute partials for alpha or proton into proton

          IF (K.EQ.1) C(1,K)=0.
          IF (K.EQ.2) C(1,K)=2.0*TOTAL(K)

C          CS(1,K)=C(1,K)
C          CS(2,K)=C(2,K)
C
C     Merge total cross sections in
          C(K,K)=C(K,K)-TOTAL(K)
          CS(K,K)=CS(K,K)+TOTAL(K)
C     Output results for current energy
C
        END DO
        WRITE(13,200) ((C(K,I),K=IZLO,IZUP),I=IZLO,IZUP)

C
C     Nuclear stopping power table:   Sept. 1993
        DO K=IZLO,IZUP
           ETOTL(K)=0.
           DO I=IZLO,IZUP
             ETOTL(K)=ETOTL(K)+ENLOSS(K,I)*CS(K,I)
           END DO
        END DO
C
        WRITE(17,200)(ETOTL(K),K=IZLO,IZUP)
C
      END DO
C
C     Close output file and stop
      CLOSE(UNIT=13)
      CLOSE(UNIT=17)
      RETURN
      END
```

```
      SUBROUTINE SPLINE( X,Y, N,NMAX, NATURAL,YP1,YPN, Y2,U )
C
C     Given arrays X and Y of length N containing a tabulated function
C     Y(i) = f[X(i)] specified upon an set of montonically increasing
C     arguments {X(1)<X(2)<...<X(N)}, SPLINE produces the array Y2
C     containing the second derivative of the interpolating function at
C     the same arguments.  The boundary conditions are specified by the
C     logical array
C
C             NATURAL(1) = .TRUE.  : Y2(1) = 0.0
C             NATURAL(1) = .FALSE. : Y2(1) = YP1
C
C             NATURAL(2) = .TRUE.  : Y2(N) = 0.0
C             NATURAL(2) = .FALSE. : Y2(N) = YPN.
C
C     Note that dummy arguments YP1 and YPN must be supplied even if
C     not used.
C
C     The algorithm uses a scratch array U, which is included in the
C     argument list so that its dimensions need not be adjusted for
C     each application.  The maximum dimension for the input and
C     output arrays is NMAX.
C
C     Adapted from "Numerical Recipes", by W.H. Press et al.
C
CX      IMPLICIT REAL*8 (A-H, O-Z)
      IMPLICIT REAL*4 (A-H, O-Z)   !for use with minfun
      LOGICAL NATURAL(2)
      DIMENSION X(NMAX), Y(NMAX), Y2(NMAX), U(NMAX)
C
C      ... Check arguments and boundary conditions.
C
      IF (N .GT. NMAX) STOP ' Too many points in SPLINE.'
C
      IF ( NATURAL(1) ) THEN
        Y2(1) =  0.0
        U (1) =  0.0
      ELSE
        Y2(1) = -0.5
        U (1) = (3.0 / (X(2) - X(1)))*((Y(2) - Y(1))/
     .                 (X(2) - X(1)) - YP1)
      END IF
C
      IF ( NATURAL(2) ) THEN
        QN = 0.0
        UN = 0.0
        Y2(N) =  0.0
        U (N) =  0.0
      ELSE
        QN = 0.5
        UN = (3.0/(X(N) - X(N-1)))*(YPN - (Y(N) - Y(N-1))/
     .            (X(N) - X(N-1)))
      END IF
C
C     ... Decomposition loop of tridiagonal algorithm.  Y2 and U are used
C         for temporary storage of decomposed factors.
C
      DO I = 2, N-1
        SIG   = (X(I) - X(I-1)) / (X(I+1) - X(I-1))
        P     = SIG * Y2(I-1) + 2.0
```

```
          Y2(I) = (SIG - 1.0) / P
          U (I) = (6.0 * ((Y(I+1) - Y(I)   ) / (X(I+1) - X(I) ) -
     .                     (Y(I)    - Y(I-1)) / (X(I)    - X(I-1))) /
     .                   (X(I+1) - X(I-1)) - SIG*U(I-1)) / P
        END DO
        Y2(N) = (UN - QN*U(N-1)) / (QN*Y2(N-1) + 1.0)
C
C       ... Backsubstitution loop.
C
        DO K = N-1, 1, -1
          Y2(K) = Y2(K)*Y2(K+1) + U(K)
        END DO
C
        RETURN
        END
        SUBROUTINE SPLINT( XA,YA,Y2A, N, X,Y )
C
C       Given arrays XA and YA of length N, which tabulate a function
C       (with the XA(i)'s in order), and given the array Y2A, which is
C       output from SPLINE above, and given a value of X, this routine
C       returns a cubic-spline interpolated value Y.
C
C       Adapted from "Numerical Recipes", by W.H. Press et al.
C
cx        IMPLICIT REAL*8 (A-H, O-Z)
        IMPLICIT REAL*4 (A-H, O-Z)   !for use with minfun
        DIMENSION XA(N), YA(N), Y2A(N)
C
C       ... Locate nearest base points by bisection.
C
        KLO = 1
        KHI = N
      1 IF ( (KHI - KLO) .GT. 1) THEN
          K = (KHI + KLO) / 2
          IF (XA(K) .GT. X) THEN
            KHI = K
          ELSE
            KLO = K
          END IF
          GO TO 1
        END IF
        H =  XA(KHI) - XA(KLO)
        IF (H .EQ. 0.0) STOP ' Arguments for SPLINE must be unique.'
C
C       ... Evaluate cubic spline polynomial.
C
        A = (XA(KHI) - X       ) / H
        B = (X       - XA(KLO)) / H
        Y = A*YA(KLO) + B*YA(KHI)
     .   +  ( A*(A*A - 1)*Y2A(KLO)   + B*(B*B - 1)*Y2A(KHI) )*H*H / 6.0
C
        RETURN
        END
```

```
      REAL FUNCTION CUT96(IZ,EN,YEAR,IMODE)
C
C     THIS ROUTINE OBTAINS DIFFERENTIAL PARTICLE FLUXES AND
C     APPLIES THE GEOMAGNETIC CUTOFF TRANSMISSION FUNCTION
C     AND RETURNS THE RESULTING FLUX, MODULATED TO THE
C     ORBIT-AVERAGE CUTOFF.
C
C     IZ = ION ATOMIC NUMBER.
C     EN = ION ENERGY IN MEV/AMU.
C     YEAR = YEAR (1975.144 = SOLAR MIN.; 1980.598 = SOLAR MAX.).
C
      IMPLICIT NONE
      INTEGER*4 IZ,IQ,IMODE,KZ,IDUM,J
      REAL*4 EN,YEAR,A,SEP_QSTATES,AN,Q,P,TRF
      REAL*4 MAGNETIC_RIGIDITY,GET_GTF,GCR_FLUX,ACR_FLUX,SEP_FLUX
      REAL*4 GCRF,ACRF,ACRFQ,SF
      COMMON/MASS/A(109)
      COMMON/SEP_QSTATES/SEP_QSTATES(30,30)

      CUT96=0.0
      IF (IMODE.LT.0 .or. IMODE.GT.3) RETURN
      IF (EN.LT.0.) RETURN
      IF (IZ.LT.1 .or. IZ.GT.92) RETURN

      AN=A(IZ)

      IF (IMODE.EQ.0) THEN

C        Galactic-Cosmic Ray Component
         Q=IZ*1.0
         P=MAGNETIC_RIGIDITY(EN,Q,AN)
         TRF=GET_GTF(P)
         GCRF=GCR_FLUX(IZ,EN,YEAR,IDUM)
         CUT96=CUT96+GCRF*TRF

C        Anomalous Component
         ACRF=0.0
         DO 100 IQ=1,IZ
         ACRFQ=ACR_FLUX(IZ,IQ,EN,YEAR)
         IF (ACRFQ.GT.0) THEN
            Q=IQ*1.0
            P=MAGNETIC_RIGIDITY(EN,Q,AN)
            TRF=GET_GTF(P)
            ACRF=ACRF+ACRFQ*TRF
         ENDIF
 100     CONTINUE

         CUT96=CUT96+ACRF

      ELSEIF (IMODE.NE.0) THEN

C        Solar Energetic Particle Contribution

         SF=SEP_FLUX(IZ,EN,IMODE)
         IF (SF.EQ.0.) RETURN
         KZ=IZ

C        For elements heavier than Zn, use Zn charge states
         IF (KZ.GT.30) KZ=30
```

```
          DO 3200 J=1,KZ
             TRF=0.0
             IF (SEP_QSTATES(KZ,J).GT. 1.0E-8) THEN
                 Q=J*1.0
                 P=MAGNETIC_RIGIDITY(EN,Q,AN)
                 TRF=GET_GTF(P)
             ENDIF
          CUT96=CUT96+SF*SEP_QSTATES(KZ,J)*TRF
3200      CONTINUE

      ENDIF

      RETURN
      END
```

```
      BLOCK DATA D01
C***********************************************************************
C     Atomic Mass Tabulation from Review of Particle Properties
C                                    Physics Letters B204 (April, 1988)
C***********************************************************************
      COMMON/MASS/AMASS(109)
      DATA AMASS/
     & 1.00794,4.002602,6.941,9.012182,10.811,12.011,14.00674,15.9994,
     & 18.9984032,20.1797,22.989768,24.305,26.981539,28.0855,30.973762,
     & 32.066,35.4527,39.948,39.0983,40.078,44.95591,47.88,50.9415,
     & 51.9961,54.93805,55.847,58.9332,58.69,63.546,65.39,69.723,
     & 72.61,74.92159,78.96,79.904,83.80,85.4678,87.62,88.90585,91.224,
     & 92.90638,95.94,98.,101.07,102.9055,106.42,107.8682,112.411,
     & 114.82,118.71,121.75,127.6,126.90447,131.29,132.90543,137.327,
     & 138.9055,140.115,140.90765,144.24,145,150.36,151.965,157.25,
     & 158.92534,162.5,164.93032,167.26,168.93421,173.04,174.967,
     & 178.49,180.9479,183.85,186.207,190.2,192.22,195.08,196.96654,
     & 200.59,204.3833,207.2,208.98037,209.,210.,222.,223.,226.0254,
     & 227.0278,232.0381,231.03588,238.0289,237.0482,244.,243.,247.,
     & 247.,251.,252.,257.,258.,259.,260.,261.,262.,263.,262.,265,266/
      END
```

```
      REAL*4 FUNCTION DIFPLD(S,L,W,H)
C
C          THIS FUNCTION RETURNS THE PROBABILITY DENSITY FOR PATHS
C     OF LENGTH S THROUGH A PARALLELEPIPED OF DIMENSIONS
C     L, W, AND H.  S, L, W, AND H MUST BE IN THE SAME UNITS.
C          THIS IS AN EXACT SOLUTION, DUE TO M. D. PETROFF OF
C     ROCKWELL INTERNATIONAL (SEE J. C. PICKEL AND J. T. BLANDFORD,
C     IEEE TRANS. ON NUCL. SCI. NS-27, 1006(1980)) WITH
C     SIMPLIFICATIONS DUE TO WARREN BENDEL OF NRL (PRIVATE
C     COMMUNICATION).  THE EQUATION NUMBERS REFER TO THE APPENDIX
C     OF PICKEL AND BLANDFORD'S PAPER.
C
C     Modified by AJT 4-2-96: IMPLICIT NONE and variable-type declarations
C     added
C

      IMPLICIT NONE
      REAL*4 S,L,H,W,AP,G
C
C     EQUATION (A-7)
C
      AP=3.*(H*W+H*L+L*W)
C
C     EQUATION (A-8)
C
      DIFPLD=(G(S,L,W,H)+G(S,W,L,H)+G(S,L,H,W)+G(S,W,H,L)+
     1 G(S,H,W,L)+G(S,H,L,W))/(3.1416*AP)
      RETURN
      END

      REAL*4 FUNCTION G(S,X,Y,Z)
      IMPLICIT NONE
      REAL*4 S,X,Y,Z,KSQ,T,RSQ,R,V,PSQ,QSQ,TSQ
C
C     PRELIMINARY DEFINITIONS
C
      KSQ=X*X+Y*Y
      TSQ=X*X+Z*Z
      T=SQRT(TSQ)
      RSQ=KSQ+Z*Z
      R=SQRT(RSQ)
      V=12.*X*Y*Z*Z
      PSQ=S*S-Z*Z
      QSQ=S*S-X*X-Z*Z
      IF((S.GE.0.0).AND.(S.LT.Z)) GO TO 10
      IF((S.GE.Z) .AND.(S.LT.T))GO TO 20
      IF((S.GE.T) .AND.(S.LE.R))GO TO 30
      G=0.0
      RETURN
C
C     EQUATION (A-9)
C
 10   G=8.*Y*Y*Z/KSQ-S*(3.*X*Y/(R*T))**2
      RETURN
C
C     EQUATION (A-10)
C
 20   G=S*(3.*Y/SQRT(KSQ))**2-S*(3.*X*Y/(T*R))**2
     1 -X*(SQRT(PSQ)/S)*(8.+4.*Z*Z/(S*S))
     2 +(V*ATAN(Y/X)-(Y*Z*Z/SQRT(KSQ))**2)/(S*S*S)
```

```
      RETURN
C
C     EQUATION (A-11)
C
 30   G=-S*(3.*X*Z/(R*SQRT(KSQ)))**2
     1 +(X*X*Z*Z*(Z*Z/KSQ-3.)+V*ATAN(Y/X))/(S*S*S)
     2 +Y*Z*Z*(SQRT(QSQ)/S)*(8./TSQ+4./(S*S))
     3 -(V/(S*S*S))*ACOS(X/SQRT(PSQ))
      RETURN
      END
```

```
      PROGRAM DOSE_DRIVER

      IMPLICIT NONE
      CHARACTER*80 INFILE,OUTFILE
      REAL*4 EMINCUT,EMAXCUT,ELOWER,EUPPER
      INTEGER*4 IZMIN,IZMAX,IZLO,IZUP,M,L
      CHARACTER*12 TARGET
      INTEGER*4 MARR,NELM,LARR
      PARAMETER (MARR=5000,NELM=92,LARR=1002)
      REAL*4 INPUT_FLUX(NELM,MARR),LETFLUX(LARR)
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE
      REAL*4 LETMINMG,LETMAXMG,LETMIN,LETMAX
      REAL*4 DOSE_PER_SECOND, ACCUMULATED_DOSE
      INTEGER*4 MODEL_TYPE
C
C     Get parameters of dose calculation:
C
      CALL INIDOSE(INFILE,LETMINMG,LETMAXMG,
     *             IZMIN,IZMAX,EMINCUT,EMAXCUT,
     *             TARGET,OUTFILE)

C     Unload input particle flux file into array:
C
      CALL UNLOAD_PARTIAL_FLUX(INFILE,IZMIN,IZMAX,EMINCUT,EMAXCUT,
     *                         ELOWER,EUPPER,M,IZLO,IZUP,
     *                         INPUT_FLUX)
C

C     Check model-type

      CALL GET_CREME96_FLUX_MODEL(INFILE,MODEL_TYPE)
C
C     Now do integral LET spectrum calculation:
C
      LETMIN=LETMINMG*1000.0
      LETMAX=LETMAXMG*1000.0
      L=LARR

      CALL ULET96(LETMIN,LETMAX,TARGET,
     *            ELOWER,EUPPER,M,IZLO,IZUP,
     &            INPUT_FLUX,L,LETFLUX)
C
C
C     Now do numerical integration to get dose value:
C         DOSE_PER_SECOND = average dose rate (rads/second)
C         ACCUMULATED_DOSE = krad or krad/sec, depending on MODEL_TYPE
C
      CALL CREME96_DOSE(L,LETMIN,LETMAX,LETFLUX,MODEL_TYPE,
     &                  VERSION_NUMBER,PROGRAM_CODE,
     &                  DOSE_PER_SECOND,ACCUMULATED_DOSE)
C
C     Now write dose results to output file:
C
      CALL OUTPUT_CREME96_DOSE(INFILE,IZLO,IZUP,LETMIN,LETMAX,
     *                         EMINCUT,EMAXCUT,TARGET,MODEL_TYPE,
     *                         VERSION_NUMBER,PROGRAM_CODE,
     *                         DOSE_PER_SECOND,ACCUMULATED_DOSE,
     &                         OUTFILE)
C
C
```

```
      STOP
      END
```

```
      SUBROUTINE EVALUATE_SEU_CROSS_SECTION
     &           (EN,NPTS,IPARAM,PARAMS,XSECT_FILE,XSECT)
C
C     Subroutine to evaluate SEU cross-section for an array
C     of abscissa values:
C
C     This same routine is used for both proton-induced and heavy-ion
C     induced cross-sections; but the dimensions of the inputs and
C     output are different in the two cases.
C
C
C     INPUTS: EN:     array of proton energies (in MeV) for proton SEUS
C                  OR array of LET values (in MeV-cm2/mg) for heavy-ion SEUs
C             NPTS:   number of points in the array
C             IPARAM: specifies cross-section model or format:
C                     IPARAM=0:  table of values
C                     IPARAM=1:  Bendel 1-parameter fit
C                     IPARAM=2:  Bendel 2-parameter fit
C                     IPARAM=4:  Weibull fit
C             PARAMS: array of at least dimension IPARAM, containing
C                     the fit parameters for IPARAM=1,2, or 4.
C
C             XSECT_FILE: name of file containing cross-section table
C                          (for IPARAM=0 option).  Cross-section values
C                          will be linearly interpolated in this table,
C                          with zero below the first entry's abscissae
C                          and a plateau  value at the the last entry's
C                          ordinate
C    OUTPUT:  XSECT:   array containing the cross-section values
C                      corresponding to values in EN array
C                      in 1.0E-12 cm2/bit (for proton cross-sections)
C                 OR   in 1.0E-8 cm2/bit (for heavy ion cross-sections)
C
C    Written by:     Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C    Last update:    29 March 1996
C
C------------------------------------------------------------------------
C
      IMPLICIT NONE
      INTEGER*4 IPARAM,K,NPTS,NSV,NSVMAX
      REAL*4 EN,XSECT,PARAMS,A,B,O,W,P,BENDEL1,BENDEL2,WEIBULL
      REAL*4 XV,YV
      REAL*4 INTERPOLATE_XSECT_TABLE
      CHARACTER*80 XSECT_FILE
      PARAMETER (NSVMAX=5000)
      DIMENSION XV(NSVMAX),YV(NSVMAX)
      DIMENSION EN(1),XSECT(1),PARAMS(4)

      IF (IPARAM.EQ.0) THEN
          CALL UNLOAD_XSECT_FILE(XSECT_FILE,NSV,XV,YV)
      ELSEIF (IPARAM.EQ.1) THEN
          A=PARAMS(1)
      ELSEIF (IPARAM.EQ.2) THEN
          A=PARAMS(1)
          B=PARAMS(2)
```

```
      ELSEIF (IPARAM.EQ.4) THEN
          O=PARAMS(1)
          W=PARAMS(2)
          P=PARAMS(3)
          A=PARAMS(4)
      ELSE
          WRITE(6,9999) IPARAM
9999      FORMAT('@ 10001 ABNORMAL TERMINATION: ',
     &    /,1x,' ERROR in EVALUATE_SEU_CROSS_SECTION: ',
     &    /,1x,' CROSS-SECTION STEERING CODE UNKNOWN: ',I5,
     &    /,1x,' STOP.')
          STOP
      ENDIF

      IF (NPTS.LE.0) RETURN
      DO 1000 K=1,NPTS
         XSECT(K)=0.0
         IF (IPARAM.EQ.0) THEN
             XSECT(K)=INTERPOLATE_XSECT_TABLE(NSV,XV,YV,EN(K))
         ELSEIF (IPARAM.EQ.1) THEN
             XSECT(K)=BENDEL1(A,EN(K))
         ELSEIF (IPARAM.EQ.2) THEN
             XSECT(K)=BENDEL2(A,B,EN(K))
         ELSEIF (IPARAM.EQ.4) THEN
             XSECT(K)=WEIBULL(O,W,P,A,EN(K))
         ENDIF
1000  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE E_LOSS(IZ,IA,JZ,JA,KZ,KA,ELAB,dKE,SigmaKE)
C
C     Computes the average energy loss dKE and variance SigmaKE when
C     nuclide (IZ,IA) impinges on medium (JZ,JA)
C     producing fragment (KZ,KA).
C     Fragment is no longer at energy ENERGY, i.e. straight-
C     ahead approximation is relaxed. Medium can be Hydrogen,
C     Helium, or any other nuclide.
C
C     Based on the paper by Barghouty, Tsao, and Silberberg, 23rd ICRC,
C     Calgary, Canada, 1993.
C
C     January 1994
C
C
      INTEGER AP,ZP,AT,ZT,AA,AAZ,BB,BBZ,CC,CCZ
      INTEGER AF,ZF,AFF,ZFF,DELTA_A,DELTA_Z,AC,ZC,AT0,ZT0
C

      DATA WN,EFERMI/931.504,38./
      DATA CONST,R0,PI/1.44,1.2,3.14159/
      DATA Ebin/20./
      DATA IENT/0/
C
C
      IF (IENT.EQ.0) THEN
          IENT=1
          WRITE(6,9999)
 9999     FORMAT(1x,' In nuclear transport: Subroutine E_LOSS active.')
      ENDIF
      dKE=0.
      SigmaKE=0.
C
      IF(ELAB.LT.50.) RETURN
C
      ZP=IZ
      AP=IA
      ZT=JZ
      AT=JA
      ZF=KZ
      AF=KA
C
      ZFF=KZ
      AFF=KA
C
      IF(AF.GE.AP) RETURN
C
      PLAB=SQRT((ELAB+WN)**2-WN**2)
      Pbin=(Ebin/ELAB)*Plab
C
C.....Energy loss calculation for Z(TARGET)<6:
      SCALE=1.
      IF(ZT.LT.6) THEN
         SCALE=(6.+ZT)/12.
         AT=12
         ZT=6
      END IF
C
C.....Energy loss calculation for Z(PROJECTILE)<6:
      SCALE=SCALE*1.
```

```
        IF(ZP.LT.6) THEN
           IF(ZP.EQ.1) THEN
             dKE=0.
             DP=0.
             RETURN
           ELSE
             SCALE=SCALE*(6.+ZP)/12.
             AP=12
             ZP=6
           END IF
        END IF
C
C.....Energy loss calculation for all other nuclides:
      CALL GLBR(AP,ZP,AT,ZT,AA,AAZ,BB,BBZ,CC,CCZ)
C     Projectile and target A and Z numbers:
      A0=AP
      AZ0=ZP
      B0=AT
      BZ0=ZT
      A=AA
      AZ=AAZ
      B=BB
      BZ=BBZ
      C=CC
      CZ=CCZ

C
      Q=HEAT(A0,AZ0,B0,BZ0,ELAB,A,AZ,B,BZ,C,CZ,TA,TB,TC,Temp)
C
C     Relative size of fragment to source "A":
 8    DELTA_A=A-AF
      DELTA_Z=AZ-ZF
C
C     Coulomb Barrier:
      Ec = (CONST*(AZ-1.))/(R0*(SQRT(A-1.)+1.))
C
      IF(DELTA_A.GE.1) THEN
C
C     CHECK IF FRAGMENT IS TOO SMALL TO BE A SPALLATION PRODUCT.
C     Here we make the assumption that if the fragment is too small,
C     i.e., fragment size < AP/2, it is accompanied by a heavy partner.
C     We proceed to calculate the loss of that heavy partner assuming
C     further that both partners suffer the same energy loss per nucleon.
C
        IF(AF.LT.DELTA_A) THEN
          AF=DELTA_A
          ZF=DELTA_Z
          GO TO 8
        END IF
C
        AFMASS=AF*WN+DROP(AF,ZF)
        AMASS=A*WN+DROP(A,AZ)
C       CONSERVE MASS:
        DMASS=AMASS-AFMASS
        TD = DELTA_A*(TA+ (3./5.)*EFERMI) + DELTA_Z*Ec
C       CONSERVE TOTAL ENERGY:
        TF = (AMASS+TA*A)- (DMASS+TD) - AFMASS
        PF = SQRT((AFMASS+TF)**2-AFMASS**2)
        dKE=ELAB-TF/AF
        dKE=SCALE*dKE
```

```
            IF(dKE.LT.0.) dKE=0.
            PERC=(dKE/ELAB)*100.
            DP=PLAB-PF/AF
          ELSE
           AC=AP+(AT-B)
           ZC=ZP+(ZT-BZ)
           DELTA_A=AC-AF
           DELTA_Z=ZC-ZF
C         Coulomb Barrier:
          Ec = (CONST*(ZC-1.))/(R0*(SQRT(AC-1.)+1.))
C
            AFMASS=AF*WN+DROP(AF,ZF)
            ACMASS=AC*WN+DROP(AC,ZC)
C           CONSERVE MASS:
            DMASS=ACMASS-AFMASS
            TCN=((AP-DELTA_A)*ELAB+(AT-B)*TA+DELTA_A*TA)/AC
            TD = DELTA_A*(TCN+(3./5.)*EFERMI)+DELTA_Z*Ec
C           CONSERVE TOTAL ENERGY:
            TF = (ACMASS+TCN*AC)- (DMASS+TD) - AFMASS
            PF = SQRT((AFMASS+TF)**2-AFMASS**2)
            dKE=ELAB-TF/AF
            dKE=SCALE*dKE
            IF(dKE.LT.0.) dKE=0.
            DP=PLAB-PF/AF
          END IF
C
C         Sigma in KE loss distribution:
C          SIGMAKE = Temp*SQRT(9.*(AP-AFF)/AP)*(SQRT(.667*dKE/Temp)+1.)
          SIGMAKE=(SIGMAKE/ELAB)*100.
C
          RETURN
          END
C
          FUNCTION HEAT(A0,AZ0,B0,BZ0,ELAB,A,AZ,B,BZ,C,CZ,TA,TB,TC,Temp)
C
C         CALCULATES THE ENERGY AND MOMENTUM OF THE THREE SOURCES A,B,C
C
          COMMON/RAPIDITY/YA0,YB0,YA,YB,YC
          DATA WN/931.504/
          DATA PI,EPS/3.14159,0.03/
C
C         Transport parameters;
C            X0      Energy leaked to the spectators
C            Y0      Longitudinal momentum degradation of spectators
C            Z0      Tranveres momentum transfer:
C
          DATA X0,Y0,Z0/.05,.25,60./
C
C         Sources A and Z numbers:
C         Note---These are estimated using Glauber theory. They are
C         impact-parameter averaged numbers!
C                    A is projectile spectator
C                    B is target spectator
C                and C is participant source.
C
C         Masses of sources A, B, and C:
          A0MAS=A0*WN+DROP(A0,AZ0)
          B0MAS=B0*WN+DROP(B0,BZ0)
          AMAS=A*WN+DROP(A,AZ)
          BMAS=B*WN+DROP(B,BZ)
```

```
      CMAS=C*WN+DROP(C,CZ)
C
      EA=A0MAS+A0*ELAB
      PA=SQRT(EA**2-A0MAS**2)
      BETA0=PA/EA
      GAMMA0=1./SQRT(1.-BETA0**2)
      YA0=0.5*ALOG(ABS((1.+BETA0)/(1.-BETA0)))
      E0=EA+B0MAS
C
C     In c.m. frame:
      V=PA/E0
      G=1./SQRT(1.-V**2)
      BETA_CM=V
      GAMMA_CM=G
      PA0=G*(PA-EA*V)
      E0=SQRT(E0**2-PA**2)
C
C     Transport parameters X, Y, and Z, averaged over impact parameter:
C     [Recalculated 15 Dec. 1993]
C
      T_FACTOR=(BETA_CM/BETA0)/GAMMA0**2
C
      X = (1./2.) * X0 * T_FACTOR
      Y = (1./2.) * Y0 * T_FACTOR
      Z = (PI/4.) * Z0 * T_FACTOR
C
C
C     Momenta of sources A, B, and C in the c.m. frame:
      PA= (1.-Y)*PA0*A/A0
      PB=-(1.-Y)*PA0*B/B0
      PC=-PA-PB
      PAX= Z*A
      PBX=-PAX
      PCX=-PAX-PBX
C
C     Iteration to find Q, the generated heat, conserving energy:
      N=1
      Q=0.
      DQ=0.
 10   Q=Q+DQ
         N=N+1
         WA=AMAS+A*Q*X
         WB=BMAS+B*Q*X
         WC=CMAS+C*Q*(1.+(1.-X)*(A+B)/C)
         EA=SQRT(PAX**2+PA**2+WA**2)
         EB=SQRT(PBX**2+PB**2+WB**2)
         EC=SQRT(PCX**2+PC**2+WC**2)
         DQ=(E0-EA-EB-EC)/(A0+B0)
C     CHECK AVAILABLE ENERGY
         HEAT=DQ
         IF (Q.EQ.0..AND.DQ.LT.0.) RETURN
         IF (ABS(DQ).GT.EPS) GO TO 10
         E00=EA+EB+EC
C         IF ((E0-E00).GT.((A0+B0)*EPS)) PRINT 101, E0,E00
C
C     Excitation Energy/nucleon:
      Temp=2./3.*Q*SQRT(T_FACTOR)
      HEAT=Q
         TA=(EA-AMAS)/A
         TB=(EB-BMAS)/B
```

```
          TC=(EC-CMAS)/C
C
      Q=G*(EA+PA*V)
      PA=G*(PA+EA*V)
      EA=Q
C
      Q=G*(EB+PB*V)
      PB=G*(PB+EB*V)
      EB=Q
C
      Q=G*(EC+PC*V)
      PC=G*(PC+EC*V)
      EC=Q
C
      TA=(EA-AMAS)/A
      TB=(EB-BMAS)/B
      TC=(EC-CMAS)/C
      EE=EA+EB+EC
      P=PA+PB+PC
      E00=SQRT(EE**2-P**2)
C
C      IF ((E0-E00).GT.((A0+B0)*EPS)) PRINT 101,E0,E00,EPS
C
      RETURN
C
 101  FORMAT (2X,20H ENERGY CONSERVATION,3F12.5)
 12   FORMAT(/2X,6E12.4//2X,6E12.4)
      END
C
      FUNCTION DROP (A,Z)
C
C     CALCULATES THE LYSEKIL NUCLEAR MASS DEFECT+WIGNER+PAIRING+SHIFT
C
      DATA A1,A2,C3,CAPPA/15.4941,17.9439,0.7053,1.7826/
      DATA C4/1.1533/
      DATA WA,WN,WP/931.504,8.07169,7.28922/
      DATA WIG,D1,D2,SHIFT/30.,12.,10.,50./

      DROP=0.
      IF (A.LT.0.9) RETURN
      A3=A**0.333333
      EN=(-A1*A+A2*A3**2)*(1.-CAPPA*(1.-2.*Z/A)**2)
      EC=(C3/A3-C4/A)*Z**2
      W=(A-Z)*WN+Z*WP
      DROP=W
      IF (A.LE.4.) RETURN
      E=EN+EC+W

      T=ABS(1.-2.*Z/A)
      EW=WIG*T
      IA2=A/2.+0.1
      IF (A-2.*IA2-0.1) 10,10,20
   10 IZ2=Z/2.+0.1
      EP=D1/SQRT(A)-D2/A
      IF (Z-2.*IZ2-0.1) 11,11,15
   11 EW=EW-EP
      GO TO 30
   15 EW=EW+EP
      IF (IA2.EQ.2*IZ2) EW=EW+WIG/A
      GO TO 30
```

```
   20 EW=EW+D2/A
   30 CONTINUE
      E=E+EW+SHIFT/A
      DROP=E
C
      RETURN
      END
C
C
      SUBROUTINE GLBR(AP,ZP,AT,ZT,AA,AAZ,BB,BBZ,CC,CCZ)
C
C     Calculates (average) numbers of proj. and target participants according
C     to Glauber theory, see, e.g., Tsao et al., PRC 47, 1257 (1993).
C
      INTEGER AP,ZP,AT,ZT,AA,AAZ,BB,BBZ,CC,CCZ
      INTEGER ZERO_A,ZERO_Z
C
      DATA PI,R0/3.14159,1.36/
      DATA P13,P23/0.33333,0.66667/
C
      FACTOR1 = (AP**P13+AT**P13)**2
      FACTOR2 = AP**P23+2*AP**P13*AT**P13
C
C     Participants:
        AP_P = AP * AT**P23 / FACTOR1
        AT_P = AT * AP**P23 / FACTOR1
        ZP_P = ZP * AT**P23 / FACTOR1
        ZT_P = ZT * AP**P23 / FACTOR1
C
C     Participant source "C":
        CC = NINT(AP_P)+NINT(AT_P)
        CCZ = NINT(ZP_P)+NINT(ZT_P)
C
C     Projectile spectator source "A":
        AA = AP - NINT(AP_P)
        AAZ = ZP - NINT(ZP_P)
C
C     Target spectator source "B":
        BB = AT - NINT(AT_P)
        BBZ = ZT - NINT(ZT_P)
C
C     Check baryon number conservation:
C
      ZERO_A = (AP+AT) - (AA+BB+CC)
      ZERO_Z = (ZP+ZT) - (AAZ+BBZ+CCZ)
      IF(ZERO_A.NE.0.OR.ZERO_C.NE.0) THEN
C        PRINT*,' ***Baryon Number Conservation***'
      END IF
C
C
        RETURN
        END
```

```
      PROGRAM FLUX_DRIVER
C
C     Driver program for generating CREME96 model fluxes
C
      IMPLICIT NONE
      INTEGER*4 IZMIN,IZMAX,IMODE,ITRANS,M
      REAL*4    EMIN,EMAX,YEAR
      CHARACTER*80 GTRANSFILE,TRAPDFILE,FLXFILE

      INTEGER*4 MARR,NELM,VERSION_NUMBER,PROGRAM_CODE
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 E,FLX
      DIMENSION E(MARR),FLX(NELM,MARR)

      CALL INIFLUX(IZMIN,IZMAX,EMIN,EMAX,YEAR,IMODE,ITRANS,
     *             GTRANSFILE,TRAPDFILE,FLXFILE)



      CALL CREME96_FLUX(IZMIN,IZMAX,EMIN,EMAX,YEAR,IMODE,ITRANS,
     *                        GTRANSFILE,TRAPDFILE,
     *                        VERSION_NUMBER,PROGRAM_CODE,
     *                        M,E,FLX)


      CALL OUTPUT_CREME96_FLUX(IZMIN,IZMAX,EMIN,EMAX,
     *                         YEAR,IMODE,ITRANS,
     *                         GTRANSFILE,TRAPDFILE,
     *                         VERSION_NUMBER,PROGRAM_CODE,
     *                         M,FLX,FLXFILE)

      STOP
      END
```

```
      REAL FUNCTION GCR_FLUX(IZIN,EEZ,YDUM,IPDUM)

      IMPLICIT NONE
      INTEGER I,IZIN,IPDUM,IMonthMean,Nmonths

C     Nmonths needs to be odd number to properly center on input month
      PARAMETER (Nmonths=1)

      REAL EEZ,YDUM,YEARAVG(Nmonths),GCR_MONTHLY_FLUX

C-----------------------------------------------------------------------


C     Perform average over neighboring months to smooth out GCR fluxes

      GCR_FLUX=0.0

      IF (Nmonths .LE. 1) THEN

        YearAvg(1)=YDUM
        GCR_FLUX=GCR_MONTHLY_FLUX(IZIN,EEZ,YDUM,IPDUM)

        IF (Nmonths .LT. 1) THEN
          WRITE(*,222)
222       FORMAT(1X,'Using number of months = 1 in GCR flux averages')
        ENDIF

      ELSE

        IMonthMean=(Nmonths+1)/2
        DO I=1,NMonths
          YearAvg(I)=YDUM + FLOAT(I-IMonthMean)/12.0
          GCR_FLUX=GCR_FLUX +
     &              GCR_MONTHLY_FLUX(IZIN,EEZ,YearAvg(I),IPDUM)
        ENDDO

        GCR_FLUX=GCR_FLUX/FLOAT(Nmonths)
      ENDIF

      RETURN

      END

C-----------------------------------------------------------------------
C
C
      REAL FUNCTION GCR_MONTHLY_FLUX(IZIN,EEZ,YDUM,IPDUM)

      IMPLICIT NONE

      INTEGER I,IZIN,IZ,IPDUM,IENT,K,J,Jyear
      INTEGER IYMIN,IYMAX,IY
      REAL*4 M0,EEZ,EN,R,R0,YDUM,PI,
      REAL W(12,300),WF(3600),AN(92),d(28),b(28),to(28)
      REAL A(28),DD(28),ALPHA(28),GAMMA(28)
      REAL Z,T,T0,BETA,SINE,ARG1,ARG2,DELTA,F,DRDE,FLUX
      EQUIVALENCE(W(12,300),WF(3600))
CX      DIMENSION ENVAL(100),FVAL(100)   !unused

C     For extrapolating wolf numbers to dates outside of data ranges,
```

```
C       assuming a 22 year periodicity.  PRB.

        REAL YearMin,YearMax
        REAL Tmin,SINE_TERM,SGN,DT,Tmax,SUM,WS
        INTEGER MonthMin,MonthMax,IBmin,IBmax,IB,N,STAT,CREME96_OPEN

C       made separate array for reading wolf number file.  This allows
C       input wolf numbers to start and end and any particular date,
C       and preserves the original structure with W and WF variables.

        REAL WOLFTMP(3600)
C
C       ENTER Tmax AND THE REST MASS OF A NUCLEON, M0
C
        DATA Tmax/14.5/,M0/931.162/

C
C       ATOMIC MASS TABLE
C
        DATA (AN(I),I=1,92)/1.,4.,6.9,
     1 9.,10.8,12.,14.,16.,19.,20.2,23.,24.3,27.,28.,
     2 31.,32.,35.5,39.9,39.,40.,45.,47.9,50.9,
     3 52.,54.9,55.8,58.9,58.7,63.5,65.4,69.7,72.6,
     4 74.9,79.,79.9,83.8,85.5,87.6,88.9,91.2,92.9,95.9,97.,101.,
     5 102.9,106.4,107.9,112.4,114.8,118.7,121.8,127.6,126.9,131.3,
     6 132.9,137.3,138.9,140.1,140.9,144.2,145.,150.4,152.,157.3,
     7 158.9,162.5,164.9,167.3,168.9,173.,175.,178.5,180.9,183.9,
     8 186.2,190.2,192.2,195.1,197.,200.6,204.4,207.2,209.,209.,
     9 210.,222.,223.,226.,227.,232.,231.,238./
C
C       TABLE 1 FROM NYMMIK ET AL.
C
        DATA b/28*1.2/,to/28*1982.5/
C
C       TABLE 2 FROM NYMMIK ET AL. NOTE: D IN THE TABLE IS DD HERE
C
        data A/1.,4.,6.9,9.,10.8,12.,14.,16.,19.,20.2,23.,24.3,27.,
     *  28.1,31.,32.1,35.4,39.9,39.1,40.1,44.9,47.9,50.9,52.,54.9,
     *  55.8,58.9,58.7/
        DATA DD/2.0E04,3.5E03,1.7E01,1.6E01,5.1E01,9.6E01,3.5E01,
     *  8.4E01,3.6E00,1.5E01,4.2E00,1.8E01,3.9E00,1.2E01,1.0E00,
     *  2.7E00,1.2E00,2.3E00,1.8E00,2.6E00,6.9E-01,2.5E00,1.13E00,
     *  2.1E00,1.04E00,9.2E00,8.7E-02,4.5E-01/
        DATA ALPHA/3.,3.,3.4,4.5,3.9,3.1,3.6,3.0,3.8,3.1,3.4,3.0,
     *  3.2,3.0,4.0,3.4,4.5,4.5,4.2,3.2,3.6,3.6,3.3,3.3,3.0,3.1,
     *  4.0,3.2/
        DATA GAMMA/2.75,2.75,2.70,2.90,3.00,2.75,2.90,2.70,3.00,
     *  2.75,2.90,2.70,2.80,2.65,2.95,2.70,3.00,2.90,3.00,2.75,
     *  2.90,2.95,2.90,2.85,2.70,2.60,2.75,2.60/
C
C----------------------------------------------------------------------
C
C       Routine extended to Z >28 using relative abundances from CREME,
C       applied to the Fe spectrum calculated here:
C
        REAL RCREME(92)

C       THE ELEMENTAL RATIOS HAVE BEEN EXTENDED TO URANIUM USING
C       THE HEAO-3 DATA INTERPRETED WITH CAMERON'S ABUNDANCES AND
C       THE RESULTS OF COSMIC RAY PROPAGATION CALCULATIONS
```

```
C        (BINNS ET AL., AP. J., VOL. 247, L115-L118,1981,
C        A.G.W.CAMERON, HARVARD-SMITHSONIAN CENTER FOR ASTROPHYSICS
C        PREPRINT SERIES NO. 1357, 1980, AND TSAO ET AL., PROC. OF
C        THE 17TH INTL. COSMIC RAY CONF., VOL. 9, P130-33, PARIS, 1981).
C        TABLE 7 IN CREME REPORT.
C
C        Revised 6/19/92 by AJT, using latest combined HEAO-3/Ariel
C        abundances, as reported by Binns et al. Ap.J 346,997-1009, 1989.
C        NOTE:  since these measurements cannot always resolve individual
C               elements, the numbers here preserve the even-odd and
C               intra-group ratios from CREME IV.
C
         DATA (RCREME(I),I=29,92)/
     1 6.8E-4,8.8E-4,6.5E-5,1.4E-4,8.9E-6,5.2E-5,
     1 9.7e-6,2.7E-5,8.8E-6,2.9E-5,6.5E-6,1.6E-5,2.9E-6,8.1E-6,9.5E-7,
     2 3.1e-6,1.6E-6,4.6E-6,1.5E-6,4.0E-6,8.8E-7,4.7E-6,9.9E-7,5.7E-6,
     3 1.1e-6,2.7e-6,6.5E-7,6.7E-6,6.0E-7,1.8E-6,4.3E-7,1.6E-6,1.9E-7,
     4 1.8e-6,3.1e-7,1.4E-6,3.5E-7,1.4E-6,5.3E-7,8.8E-7,1.8E-7,8.9E-7,
     5 1.3e-7,8.1e-7,7.3E-8,8.1E-7,2.8E-7,1.2E-6,7.9E-7,1.5E-6,2.8E-7,
     6 4.9e-7,1.5e-7,1.4E-6,7.3E-8,0.,0.,0.,0.,0.,0.,8.1E-8,0.,4.9E-8/

C-----------------------------------------------------------------------------
C

         IZ=IZIN
         IF (IZ.GT.28) IZ=26

         IF (IENT.EQ.0) THEN
         IENT=1

         d(1)=0.012
         do i=2,28
         d(i)=d(1)*an(i)/float(i)
         end do

C        CALCULATE PI
C
         PI=4.0*ATAN(1.0)
C
C        OPEN FILE OF MONTHLY AVERAGE WOLF NUMBERS FROM MCKINNON AT NOAA
C        AND READ THEM IN
C

c        use file starting in 1950

c         OPEN(UNIT=60,READONLY,SHARED,STATUS='OLD',
c     *        FILE='CREME96:WOLF.DAT')
         stat = creme96_open('wolf.dat','cr96tables',60,'old')

         IF (ydum .LT. 1950) THEN
           WRITE(*,111)
111        FORMAT(1X,'Warning, GCR results are unreliable before 1950')
         ENDIF


C        Modified read, where K is set to maximum array size.  At present,
C        this is 3600, allowing specification of 300 years.

         DO K = 1,3600
```

```
C         For determining bounds of Wolf number file.  If entered date is
C         outside of these bounds, the wolf numbers are extrapolated
C         assuming a 22 year periodicity.

          MonthMax=I
          YearMax=FLOAT(Jyear) + FLOAT(MonthMax-1)/12.0

          READ(60,1,END=2)JYEAR,I,WOLFTMP(K)
          J=(JYEAR-1749)+1
          W(I,J)=WOLFTMP(K)

          IF (K .EQ. 1) THEN
            MonthMin=I
            YearMin=FLOAT(Jyear) + FLOAT(MonthMin-1)/12.0
            MonthMax=I
            YearMax=FLOAT(Jyear) + FLOAT(MonthMax-1)/12.0

          ENDIF

        ENDDO

1       FORMAT(I4,1X,I2,F6.1)
 2      CONTINUE

        CLOSE(60)
C

        ENDIF     !IENT = 0 OPTION


C
C       COMPUTE THE BOUNDS OF THE WOLF NUMBER ARRAY.  This update
C       is to be used in smoothing algorithm, so that check if out of
C       bounds.  Also used by CheckDates routine, in order to handle
C       transition in wolf number array at the center of each month,
C       e.g. wolf number changes from month 6 (June) to month 7 (July)
C       on the 15 June, as used in IB index.
C
        IYMIN=INT(YearMin)
        IBMIN=(IYMIN-1749)*12+12.*(YearMin-IYMIN)+.5+1
        IYMAX=INT(YearMax)
        IBMAX=(IYMAX-1749)*12+12.*(YearMax-IYMAX)+.5+1


        Z=IZ
        GCR_MONTHLY_FLUX=0.0

C
C       CALCULATE DT(MONTH) FOR YEAR T
C
        T=YDUM

        IY=INT(T)         !   YDUM will now be used to pass the year of
                          !   interest to this function.  R. Witt 6/22/95
C
C       COMPUTE THE LOCATION, IB, OF THE WOLF NUMBER FOR TIME T IN
C       THE W ARRAY
C
        IB=(IY-1749)*12+12.*(T-IY)+.5+1
```

```
C       Routine checks if T is within Wolf number bounds.  If not, adjusts
C       T and IB assuming a periodicity of 22 years.

C       Note that period which crosses beginning year in wolf number data
C       boundary has been handled as a special case in the DO 70 and DO 71
C       loops below.

        CAll CheckDate(T,YearMin,Yearmax,IB,IBmin,IBmax)

C       VERSION OF T0 IN THE PAPER
C
        T0=1978.5
        IF(T.GE.1985.) T0=1976
C
        EN=EEZ
C
C       CONVERT EN TO RIGIDITY IN GV, AS R
C
        R=(AN(IZ)/Z)*(EN*EN+2.*M0*EN)**.5/1000.
C
C       CALCULATE DT(MONTH) FOR R IN YEAR T
C
        Tmin = 5.3/R**0.3
        SINE_TERM=sin(2.*PI*(t-t0)/22.)
        IF(SINE_TERM.GE.0.) SGN=1.
        IF(SINE_TERM.LT.0.) SGN=-1.
        DT = (Tmax+Tmin)/2.+((Tmax-Tmin)/2.)*SGN*
     1  (ABS(SINE_TERM))**(1./3.)
C
C       CALCULATE THE SMOOTHED WOLF NUMBER, WS
C
        N=DT+0.5
        SUM=0.

        DO 70 K=1,N
          IF ((IB-K) .GE. IBMIN) SUM=SUM+K*WF(IB-K)
C         Extrapolate backwards using 22 year solar cycle pattern
          IF ((IB-K) .LT. IBMIN) SUM=SUM+K*
     &        WF(IB-K+12*22)
70      CONTINUE

        DO 71 K=N+1,2*N-1
          IF ((IB-K) .GE. IBMIN) SUM=SUM+(2*N-K)*WF(IB-K)
C         Extrapolate backwards using 22 year repeating pattern
          IF ((IB-K) .LT. IBMIN) SUM=SUM+(2*N-K)*
     &        WF(IB-K+12*22)
71      CONTINUE

        WS=SUM/(N*N)
C
C       COMPUTE THE MODULATION POTENTIAL
C
        R0=0.375+3E-4*WS**1.445
C
C       COMPUTE BETA
C
        BETA=SQRT(1-(EN/M0+1)**(-2))
C
C       COMMPUTE DELTA
C
```

```
      SINE=SIN(2.*PI*(T-TO(IZ))/22.)
      IF(SINE.GE.0.) SGN=1.0
      IF(SINE.LT.0.) SGN=-1.0
      SINE=ABS(SINE)

c     Inserted to avoid floating underflows.

      ARG1=-BETA*R/D(IZ)
      ARG2=-BETA*R/R0
      IF (ABS(ARG1) .LE. 1.0E-20) ARG1=0.0
      IF (ABS(ARG2) .LE. 1.0E-20) ARG2=0.0

      DELTA=5.5*ABS(1-B(IZ)*EXP(ARG1))+
     * (1.13*BETA*R/R0)*(SGN*SINE**(1./3.))*EXP(ARG2)

      IF (ABS(DELTA) .LE. 1.0E-20) DELTA=0.0  !also to avoid underflows

C
C     COMPUTE THE FLUX
C
      F=(DD(IZ)*BETA**ALPHA(IZ))/R**GAMMA(IZ)
      F=F*(R/(R+R0))**DELTA
C
C     COMPUTE dR/dE
C
      dRdE=(AN(IZ)/(Z*1000.))*(EN+M0)/((EN*EN+2.*M0*EN)**.5)
C
C     CONVERT FROM PER GV TO PER MeV/nuc
C
      FLUX=F*dRdE
      GCR_MONTHLY_FLUX=GCR_MONTHLY_FLUX+FLUX


C     Scale relative to Fe for Z > 28:
      IF (IZIN.GT.28) GCR_MONTHLY_FLUX=GCR_MONTHLY_FLUX*RCREME(IZIN)
      RETURN
      END

C----------------------------------------------------------------------


      SUBROUTINE CheckDate(Year,YearMin,Yearmax,IB,IBmin,IBmax)

C     Routine checks if T is within Wolf number bounds.  If not, adjusts
C     T to being in bounds assuming a 22.0 year periodicity.
C
C     YearMin & YearMax are unused in this routine, but are included
C     as arguments for possible future use.

      IMPLICIT NONE
      REAL Year,YearMin,Yearmax
      INTEGER Ncycle,IBmin,IBmax,IB,IBlow,IBhigh,IBnew

C     fix wolf number range for extrapolating to be July 1970 to JUNE 1992
C     this algorithm assumes a 22 year periodicity.
      DATA IBlow,IBhigh/2659,2922/

C----------------------------------------------------------------------

      IF ((IB .GE. IBmin) .AND. (IB .LE. IBmax)) THEN
```

```
C          Dates are o.k., don't need to adjust YEAR or IB
           RETURN
        ELSEIF (IB .GT. IBmax) THEN
           IBnew=IBlow+MOD(IB-IBhigh-1,IBhigh-IBlow+1)
           Ncycle=(IB-IBhigh-1)/(IBhigh-IBlow+1)+1
           Year=Year-Ncycle*22.0
           IB=IBnew
        ELSEIF (IB .LT. IBmin) THEN
           IBnew=IBhigh-MOD(IBlow-IB-1,IBhigh-IBlow+1)
           Ncycle=(IBlow-IB-1)/(IBhigh-IBlow+1)+1
           Year=Year+Ncycle*22.0
           IB=IBnew
        ENDIF

        RETURN
        END
```

```
      SUBROUTINE Geomag96(OrbIncl,Apogee,Perigee,AscNodeLong,
     #              AscNodeDisp,PerigDisp,Zenith,Azimuth,UTtimeInit,
     #              Stormy,Shadow,PreCalcGTFs,IPreCalc,
     #              RigBins,TransFunc,Year,XLbounds,ILbins)

      IMPLICIT NONE


      INTEGER J,Jmax,L,Ndays,NorbSteps,IPreCalc,Nrigs,NLvals

      PARAMETER (Ndays=7,NorbSteps=200,Nrigs=1001,NLvals=10)


C     Now REAL to properly handle omnidirectional averaging.
C     Only the Earth's geometric shadow is included in the generic
C     omnidirectional averaging at present.

      REAL MAT(Nrigs,NLvals),TransInc

      INTEGER IDEX

      DATA TransInc/1.0/

      REAL RigBins(Nrigs),TransFunc(Nrigs,NLvals)

      LOGICAL Shadow,Stormy,PreCalcGTFs,GridInit,INIGRID

C     Initial Orbital & lookout direction input parameters set
C     in GTFDriverInput
      REAL OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp
      REAL Zenith,Azimuth,C,Cgrid,Csupress,DeltaNymmik
      REAL UTtimeInit,UTtime,TimeLocal
      REAL Time,Period,Step

C     Parameters along each orbital step
      REAL Zlat,Zlon,Alt

      INTEGER ILbins,ILbin,ICODE,NperLbin(NLvals)
      REAL Year,XLval,BB0,XLbounds(NLvals),XLinfinite
      PARAMETER (XLinfinite=1.0E+06)

      REAL Grid80Lval,RatioL
      LOGICAL UseLapprox

C-----------------------------------------------------------------------

C     Initializations

      DO L=1,NLvals
       NperLbin(L)=0
       DO J=1,Nrigs
         MAT(J,L)=0.0
       ENDDO
      ENDDO

      TransInc=1.0

C     Choice of original geomagnetic storm option or pre-calculated GTF
C     option.  These are mutually exclusive now.  Note that "Stormy" applies
C     to updated Grid, and thus will be applied on top of the Nymmik
```

```
C       correction for high inclination orbits if that option is chosen.


        IF (PreCalcGTFs) THEN

C         NOTE:  The pre-calculated GTFs have not been divided into L-bins.
C                This may be a useful option to include in future updates.

          CALL GetPreCalcGTF(IPreCalc,RigBins,TransFunc)

          RETURN  !could just use subsequent RETURN, since this IF statement
                  !skips all lines before the subsequent RETURN

        ELSE  !calculate GTF if not using pre-calculated ones

C         Initialize Orbit routine

          CALL Orbit(1,Period,ZLon,ZLat,Alt,Apogee,Perigee,OrbIncl,
     #               AscNodeLong,AscNodeDisp,PerigDisp)
C
C         Initialize cutoff grid.
C
          IF (.NOT. GridInit) GridInit=INIGRID

C         Compute the total number of steps in "Ndays" days if we make
C         "Norbsteps"  steps per orbit.  Use 2 days and 200 steps per orbit
C         presently.
C
          JMAX=INT(Ndays* NorbSteps*86400./PERIOD + 1.5)
C
C         Compute the step size in seconds.
C
          STEP=PERIOD/FLOAT(NorbSteps)
C
C         Compute the vertical cutoff at the spacecraft
C         position for every time step.
C

         DO J=1,JMAX

          time=FLOAT(j-1)*step

          CALL Orbit(2,Time,ZLon,ZLat,Alt,Apogee,Perigee,OrbIncl,
     #               AscNodeLong,AscNodeDisp,PerigDisp)

C       Now calculate geomagnetic cutoff from the Grid.  Perform before
C       L-value calculation, and see similarity

          CALL GET_CUTOFF(ZLAT,ZLON,ALT,Azimuth,Zenith,C)
          Cgrid=C

          IF ( XLbounds(2) .LT. XLinfinite .OR.
     &         ( XLbounds(2) .GE. XLinfinite .AND.
     &           XLbounds(1) .GT. 0.0 ) ) THEN

            CALL GridApproxLval(Cgrid,XLbounds,ILbins,Grid80Lval,
     &              UseLapprox)

            IF (UseLapprox) THEN
              XLval=Grid80Lval
```

```
          ELSE
            CALL GET_BLCOORDS(Year,Zlat,Zlon,Alt,XLval,BB0,ICODE)
          ENDIF

          IF (XLval .GT. 99999.0) XLval=99999.0

          CALL GetLbin(XLval,XLbounds,ILbins,ILbin)

          IF (ILbin .GE. 1 .AND. ILbin .LE. NLvals)
     &        NperLbin(ILbin)=NperLbin(ILbin)+1

        ELSE

C         If no L-bins are specified or 1 L-bin is specified
C         and the lower bound is L = 0, use only the first
C         element of the array.  In this case, the following
C         sum should equal JMAX once the stepping through the
C         orbit is completed.

          ILbin=1
          NperLbin(Ilbin)=NperLbin(Ilbin)+1
        ENDIF


        CALL ConvertTime(time,UTtimeInit,UTtime,Zlon,Period,
     #                   TimeLocal)

C----------------------------------------------------------------------------------
C

       IF (Cgrid .GT. 0.0) THEN

        CALL Nymmik(C,TimeLocal,DeltaNymmik)
        C=C/(1+DeltaNymmik)

        IF (Stormy) THEN
C           Now apply cutoff suppression during large magnetic storms,
C           as described by Adams, et al. (1981).
          Csupress = .54*EXP(-Cgrid/2.9)
          C=C-Csupress
          IF (C .LT. 0.) C=0.0   !lowest cutoffs are defined to be 0

        ENDIF             !applying Stormy correction

       ENDIF              !checking that Cgrid (grid cutoff) > 0.0

C
C      Histogram cutoffs in 0.02 GV steps.  Since only allow vertical and
C      western cutoffs, all IDEX should be in bounds, since C_vert < 20 GV.
C      Note that the transmission function is an integral spectra of
C      cutoffs < Rigidity.  See CALCULATE_TRANS_FUNC for algorithm
C      which assigns rigidities for bins.

       IF (C .EQ. 0.0) THEN
         IDEX=1
       ELSE
         IDEX=INT(C*50.)+2
       ENDIF

C      This is a correction for the earth's shadow on the spacecraft
```

```
C       according to simple geometrical optics.  Have made MAT real, and
C       apply to each point in calculation, in order to handle correction
C       properly for non-circular orbits.  This routine has been designed
C       to always apply the Earth's shadow, although the technique will likely
C       be modified before 1997.
C

         IF(Shadow) THEN
            TransInc=(1.-0.5*(1.-((6371.2+ALT)**2.
     1                      -(6371.2)**2.)**.5/(6371.2+ALT)))
         ENDIF  !applying Earth's shadow correction

         IF (ILbin .GE. 1 .AND. ILbin .LE. NLvals)
     &        MAT(IDEX,ILbin)=MAT(IDEX,ILbin)+TransInc

         ENDDO    !for number of orbital steps

C
C       Now calculate transmission function.
C
         CALL CALCULATE_TRANS_FUNC(Jmax,MAT,RigBins,NperLbin,TransFunc)

        ENDIF   !for using either pre-calculated GTF or GRID-based options

        RETURN
        END


C-----------------------------------------------------------------------

        SUBROUTINE GET_CUTOFF(ZLAT,ZLON,ALT,AZ,ZE,C)
C
C       For input ZLAT,ZLON,ALT,AZ,ZE, calculates cutoff C (in GV)
C           ZLON = geocentric longitude of spacecraft position (deg)
C           ZLAT = geocentric latitude of spacecraft position  (deg)
C           ALT  = spacecraft altitude (km)
C           AZ   = azimuth of particle wrt spacecraft (deg)
C           ZE   = zenith angle of particle wrt spacecraft (deg)
C
C
C       Routine modified 3/5/90:
C         In JHA's original version of this coding, he first calculated the
C            vertical cutoff at the 4 grid corners at 20 km altitude,
C            re-scaled via Stormer theory to orbital altitude and orientation,
C            and then averaged the four.  This procedure involved 5 calls
C            to subroutine FUNCTION STORMER.
C
C         In the modified coding, the 20 km vertical cutoffs are first averaged,
C            and then re-scaled via Stormer theory to orbital altitude and
C            orientation.  This procedure involves only 2 calls to STORMER.  It
C            also gives a smoother transmission function.
C
C       12/16/92, Fixed XORB bug, so that XORB is calculated for
C            all latitudes.  RGRD is now a parameter.
C
C       12/20/95, set cutoffs that are negative to 0.0.

        IMPLICIT NONE

        REAL ZLAT,ZLON,ALT,AZ,ZE,C
```

```
      REAL AZG,ZEG,RGRD

      DATA AZG/0./,ZEG/0./
      PARAMETER (RGRD=1.0031392126)  !equivalent to 6391.2/6371.2

      REAL CUTOFF(33,72),CN,CS
      COMMON/CUTOFF80/CUTOFF,CN,CS

      INTEGER ILO,IUP,JLO,JUP
      REAL ZI,ZJ,XORB,DI,DJ,SC,SG
      REAL Y1,Y2,Y3,Y4,CL
      REAL STORMER
C-----------------------------------------------------------------------


C     COMPUTE THE TABULAR POSITION OF THE VERTICAL CUTOFF.
C
C
      ZI=ZLAT/5.+17.
      ZJ=ZLON/5.+1.
      ILO=INT(ZI)
      IUP=ILO+1
      JLO=INT(ZJ)
      IF(JLO.EQ.73) JLO=1
      JUP=JLO+1
      IF (JUP.EQ.73) JUP=1
C
C     INTERPOLATE THE VERTICAL CUTOFF TO THE EXACT LOCATION
C     OF THE SPACECRAFT USING STORMER THEORY.
C
      XORB=(6371.2+ALT)/6371.2

      IF(ABS(ZLAT).GE.80.) GO TO 100
      DI=ZI-FLOAT(ILO)
      DJ=ZJ-FLOAT(JLO)

      SC=STORMER(ZLAT,ZLON,XORB,AZ,ZE)
      SG=STORMER(ZLAT,ZLON,RGRD,AZG,ZEG)

C     Vertical cutoffs
      Y1=CUTOFF(ILO,JLO)
      Y2=CUTOFF(IUP,JLO)
      Y3=CUTOFF(ILO,JUP)
      Y4=CUTOFF(IUP,JUP)
      C=(1.-DI)*(1.-DJ)*Y1+(1.-DI)*DJ*Y3+DI*(1.-DJ)*Y2+DI*DJ*Y4

      C=SC*C/SG
C-----------------------------------------------------------------------
      GO TO 200
C
C     FOR ABS(LATITUDE).GT.80 USE THE CUTOFFS AT THE POLE INSTEAD OF
C     THE CUTOFFS AT FOUR NEARBY LOCATIONS.
C
 100  CONTINUE
      DJ=ZJ-FLOAT(JLO)
      SC=STORMER(ZLAT,ZLON,XORB,AZ,ZE)
      SG=STORMER(ZLAT,ZLON,RGRD,AZG,ZEG)
      IF(ZLAT.LE.-80.)GO TO 110
      DI=ZI-33.
      CL=DJ*CUTOFF(33,JUP)+(1.-DJ)*CUTOFF(33,JLO)
```

```
      C=(DI*CN+(2.-DI)*CL)/2.
      C=SC*C/SG
      GO TO 200
110   CONTINUE
      DI=1.-ZI
      CL=DJ*CUTOFF(1,JUP)+(1.-DJ)*CUTOFF(1,JLO)
      C=(DI*CS+(2.-DI)*CL)/2.
      C=SC*C/SG
200   CONTINUE

      IF (C .LT. 0.) C=0.0   !added 12-20-95

      RETURN
      END


C-----------------------------------------------------------------------

      SUBROUTINE GetLbin(XLval,XLbounds,ILbins,ILbin)

      IMPLICIT NONE
      INTEGER ILbins,ILbin,NLvals,L
      PARAMETER (NLvals=10)
      REAL XLval,XLbounds(NLvals)
      LOGICAL FindLbin

C     No attempt is made to eliminate "unphysical" or "approximate"
C     L-values using the ICODE returned from GET_BLCOORDS, since any
C     analyses using L-values are likely to handle these locations
C     "as is", i.e. with the calculated L-value.

C-----------------------------------------------------------------------

      FindLbin=.TRUE.

      ILbin=0

      DO L=1,ILbins
       IF (FindLbin) THEN

         IF (L .LT. NLvals) THEN
           IF ( (XLval .GE. XLbounds(L)) .AND.
     &       (XLval .LT. XLbounds(L+1)) )THEN
             ILbin=L
             FindLbin=.FALSE.
           ENDIF

         ELSE  !special handling of L=NLvals case

           IF (XLval .GE. XLbounds(L))THEN
             ILbin=L
             FindLbin=.FALSE.
           ENDIF
         ENDIF   !checking of each L-bin

       ENDIF     !for FINDLbin logical
      ENDDO

      RETURN
      END
```

```
C-----------------------------------------------------------------------

C       My attempt to make this into a more modern FUNCTION, including the use
C       of IMPLICIT NONE.  5-7-96, PRB.

        REAL FUNCTION STORMER(GCLATD,GCLOND,RGC,AZ,ZE)

C
C       WE DID NOT WRITE THIS SUBROUTINE.  WE HAVE MADE NO CHANGES IN
C       IT IN 1984.
C
C
C       May 1996 comments and status, PRB.
C
C       1.  Note that this FUNCTION uses the 1975 IGRF field plus drifts.
C       In principle, the 1980 IGRF/DGRF coefficients would be more
C       appropriate, since the STORMER corrections are applied to the
C       1980 grid.  In future years, we intend to replace the GRID results
C       with a 1990 grid, and will modify this routine accordingly.
C
C       2.  The coefficients are also listed in inverted order, compared with
C       more recent tabulations, e.g. G01 is generally listed as G10 in more
C       recent tabulations.
C
C       3.  This routine HAS NOT been converted to IMPLICIT NONE, due to the
C       historical nature of the coding.
C

        IMPLICIT NONE

        REAL RED,EDLAT,AZM,ZEM,GAMMA

        COMMON/KARL/RED,EDLAT,AZM,ZEM,GAMMA

C       Need to determine usage of DPEC(U), 5-7-96, PRB.
        REAL DPEC,U,ZEDRTL,ZRTL

        INTEGER JDATA,NOPT
        REAL PI,RAD,PIO2,TWOPI,SQRT3,DT
        REAL G01,G02,G11,G12,H11,H12,G22,H22,H0,H0SQ
        REAL EL0,EL1,EL2,E,XEDFGC,YEDFGC,ZEDFGC,REDFGC
        REAL THETA,THETAD,PHI,PHID,CP,SP,ST,CT,CPCT,CPST,SPCT,SPST
        REAL R1ER,R1KM,ERAD,TH1RAD,TH1DEG,PH1RAD,PH1DEG
        REAL XGMED,YGMED,ZGMED,ZDEDNP,RGC,XODNP,YODNP,ZODNP,DODNP
        REAL DIFLA,PHINOF,TNOF,SGCLATD,CGCLATD,GCLATD,SGCLOND,GCLOND
        REAL CGCLOND,XGC,YGC,ZGC,GCT,SGCT,CGCT,GCROT,SGCROT,CGCROT
        REAL XRL,YRL,ZRL,XEDRL,YEDRL,ZEDRL,XEDRTL,YEDRTL,XRTL,YRTL
        REAL XEDP,YEDP,ZEDP,XODNPR,YODNPR,ZODNPR,XODNPT,YODNPT,ZODNPT
        REAL ROTM,SROTM,CROTM,ROTMD,PLAZ,AZ,TLZE,ZE,SPLAZ,CPLAZ
        REAL STLZE,CTLZE,XLD,YLD,ZLD,CA,A,SA,ADEG,XLP,YLP,ZLP,CB,B,SB
        REAL BDEG,XLPP,YLPP,ZLPP,ZLDM,XLDM,YLDM,SMALL,PAZM
        REAL XED1,YED1,ZED1,ZED2,COSLDA
C
C       THIS FUNCTION TRANSFORMS A GEOGRAPHIC LOCATION AND ARRIVAL
C       DIRECTION INTO OFFSET DIPOLE COORDINATES, THEN COMPUTES THE
C       STORMER CUTOFF IN GV AND RETURNS THE RESULT. THE OFFSET DIPOLE
C       COORDINATES ARE AVAILABLE IN THE COMMON BLOCK /KARL/.
C
C
```

```
C          GCLATD IS GEOCENTRIC LATITUDE  IN DEGREES
C          GCLONG IS GEOCENTRIC LONGITUDE IN DEGREES
C          RGC    IS RADIAL DISTANCE FROM GEOCENTER IN EARTH RADII
C          AZ     IS GEOGRAPHIC AZIMUTH
C          ZE     IS GEOGRAPHIC ZENITH
C          RED    IS RADIAL DISTANCE FROM OFFSET DIPOLE POSITION IN
C                                                 EARTH RADII
C       EDLAT IS THE GEOMAGNETIC LATITUDE IN OFFSET DIPOLE COORDINATES
C          AZM    IS GEOMAGNETIC AZIMUTH IN OFFSET DIPOLE COORDINATES
C          ZEM    IS GEOMAGNETIC ZENITH  IN OFFSET DIPOLE COORDINATES
C          GAMMA  IS GAMMA ANGLE MEASURED FROM MAGNETIC EAST
C
       DATA JDATA,NOPT/2*0/,PI,RAD,PIO2,XEDFGC,YEDFGC,ZEDFGC,CP,SP
     1 ,ST,CPCT,CPST,SPCT,SPST,XGMED,YGMED,ZGMED/16*-8000./
       DATA SMALL/1.0E-35/

c     In declaration section now, 5-96, PRB.

      DATA ERAD,THETAD,PHID,R1KM,TH1DEG,PH1DEG/6371.2,
     1 11.4354,-290.2392,450.2586,72.8278,148.7753/

C-----------------------------------------------------------------------

       IF(JDATA.EQ.77)   GO TO 10
         PI = ACOS(-1.0)
         RAD = 180.0/PI
         PIO2 = PI/2.0
         TWOPI = PI*2.0
       NOPT = 0
       SQRT3 = SQRT(3.0)
C          ENTER GEOMAGNETIC DATA,     IGRF  1975
C            SEE JGR, 81, 5163, 1976
C                                           DT IS NUMBER OF YEARS SINCE 1975
       DT = 5.0
       G01 = -30186.0  +  25.6*DT
       G02 =  -1898.0  -  24.9*DT
       G11 =  -2036.0  +  10.0*DT
       G12 =   2997.0  +   0.7*DT
       H11 =   5735.0  -  10.2*DT
       H12 =  -2124.0  -   3.0*DT
       G22 =   1551.0  +   4.3*DT
       H22 =    -37.0  -  18.9*DT
       IF(NOPT.EQ.1)PRINT 1000, G01, G02, G11, G12, G22, H11, H12, H22
C        COMPUTE POSITION OF OFFSET DIPOLE
       H0 = SQRT( G01*G01+G11*G11+H11*H11)
       H0SQ = H0*H0
       EL0 = 2.0*G01*G02+(G11*G12+H11*H12)*SQRT3
       EL1 = -G11*G02+(G01*G12+G11*G22+H11*H22)*SQRT3
       EL2 = -H11*G02+(G01*H12-H11*G22+G11*H22)*SQRT3
C       E = (EL0*G01+EL1*G11+EL2*H11)*4.0*H0SQ
       E = (EL0*G01+EL1*G11+EL2*H11)/(4.0*H0SQ)
       IF(NOPT.EQ.1)  PRINT 1011, EL0, EL1, EL2, E, H0
 1011  FORMAT(1H , 8E15.5)
C       XEDFGC = ERAD*(EL1-G11*E)/(3.0*H0SQ)
       XEDFGC =       (EL1-G11*E)/(3.0*H0SQ)
C       YEDFGC = ERAD*(EL2-H11*E)/(3.0*H0SQ)
       YEDFGC =       (EL2-H11*E)/(3.0*H0SQ)
C       ZEDFGC = ERAD*(EL0-G01*E)/(3.0*H0SQ)
       ZEDFGC =       (EL0-G01*E)/(3.0*H0SQ)
       REDFGC = SQRT(XEDFGC*XEDFGC+YEDFGC*YEDFGC+ZEDFGC*ZEDFGC)
```

```
      IF(NOPT.EQ.1)PRINT 3001, XEDFGC, YEDFGC, ZEDFGC, REDFGC
 3001 FORMAT (1H , 4F10.4,      3X, 'XEDFGC, YEDFGC, ZEDFGC, REDFGC')
 1000 FORMAT (1H , 10F13.5)
 1010 FORMAT(1H0, 8F15.5/1H ,8F15.5)
      THETA = THETAD/RAD
      PHI = PHID/RAD
      CP = COS(PHI)
      SP = SIN(PHI)
      ST = SIN(THETA)
      CT = COS(THETA)
      CPCT   = CP*CT
      CPST   = CP*ST
      SPCT   = SP*CT
      SPST   = SP*ST
      R1ER = R1KM/ERAD
      TH1RAD = TH1DEG/RAD
      PH1RAD = PH1DEG/RAD
      IF(NOPT.EQ.1)PRINT 1000,R1KM,TH1DEG,PH1DEG,R1ER,TH1RAD,PH1RAD
      XGMED = XEDFGC*CPCT -YEDFGC*SPCT -ZEDFGC*ST
      YGMED = XEDFGC*SP   +YEDFGC*CP
      ZGMED = XEDFGC*CPST -YEDFGC*SPST +ZEDFGC*CT
      IF(NOPT.EQ.1)PRINT 3002, XGMED, YGMED, ZGMED
 3002 FORMAT(1H , 3F10.4, 13X, 'XGMED, YGMED, ZGMED')
      IF(NOPT.EQ.1)PRINT 1010, CP, SP, CT, ST, CPCT, CPST, SPCT, SPST
      JDATA = 77
   10 CONTINUE
C        ITERATE TO FIND COORDINATES OF OFFSET NORTH DIPOLE AT ANY
C      LATITUDE
C     FIRST GUESS FIND OFFSET NORTH DIPOLE AT DISTANCE RGC
      ZDEDNP = RGC
  100 XODNP =  XGMED*CPCT + YGMED*SP + ZDEDNP*CPST
      YODNP = -XGMED*SPCT + YGMED*CP - ZDEDNP*SPST
      ZODNP = -XGMED*SP              + ZDEDNP*CT
      DODNP = SQRT(XODNP*XODNP + YODNP*YODNP + ZODNP*ZODNP)
      DIFLA = DODNP - RGC
      IF(ABS(DIFLA) - 1.0E-5)  120, 120, 110
  110 ZDEDNP = ZDEDNP - DIFLA
 4001 FORMAT (1H , 5X,'ODC 0, 0, 'F7.5, ' = GC X, Y, Z  OF'3F8.5,
     1 '   DODNP ='F9.5,'   DIF OF 'F9.6, '  AT LOND LAT'F10.4, F8.4)
      GO TO 100
  120 CONTINUE
      PHINOF = ATAN2(YODNP,XODNP)*RAD
      IF(PHINOF.LT.0.0)   PHINOF = PHINOF + 360.0
      TNOF = -ACOS(ZODNP/DODNP)*RAD + 90.0
      IF(NOPT.EQ.1)PRINT 4001,ZDEDNP,XODNP,YODNP,ZODNP,DODNP,DIFLA,
     1 PHINOF,TNOF
      SGCLATD = SIN(GCLATD/RAD)
      CGCLATD = COS(GCLATD/RAD)
      SGCLOND = SIN(GCLOND/RAD)
      CGCLOND = COS(GCLOND/RAD)
C        GET GEOCENTRIC X Y Z  COORDINATES
      XGC = RGC*CGCLATD*CGCLOND
      YGC = RGC*CGCLATD*SGCLOND
      ZGC = RGC*SGCLATD
      GCT = (90.0 - GCLATD)/RAD
      SGCT = SIN(GCT)
      CGCT = COS(GCT)
C        FIND X Y Z IN LOCAL COORDINATES OF X=0, Y=0, Z
C                  THE LOCAL COORDINATE Z AXIS PASSES THRU P
C                  THE LOCAL COORDINATE X,Z PLANE CONTAINS P
```

```
      GCROT = ATAN2(YGC,XGC)
      IF(NOPT.EQ.1)PRINT 2001, XGC, YGC, ZGC, GCROT
 2001 FORMAT(1H , 4F10.4,         3X, 'XGC, YGC, ZGC, GCROT')
      SGCROT = SIN(GCROT)
      CGCROT = COS(GCROT)
      XRL =  XGC*CGCROT*CGCT + YGC*SGCROT*CGCT - ZGC*SGCT
      YRL = -XGC*SGCROT      + YGC*CGCROT
      ZRL =  XGC*CGCROT*SGCT + YGC*SGCROT*SGCT + ZGC*CGCT
 2002 FORMAT(1H , 3F10.4,        13X, 'XRL, YRL, ZRL')
      IF(NOPT.EQ.1)PRINT 2002, XRL, YRL, ZRL
C       DETERMINE LOCATION OF OFFSET DIPOLE CENTER IN THESE SAME
C                                ROTATED LOCAL COORDINATES
      XEDRL =  XEDFGC*CGCROT*CGCT + YEDFGC*SGCROT*CGCT - ZEDFGC*SGCT
      YEDRL = -XEDFGC*SGCROT      + YEDFGC*CGCROT
      ZEDRL =  XEDFGC*CGCROT*SGCT + YEDFGC*SGCROT*SGCT + ZEDFGC*CGCT
      IF(NOPT.EQ.1)PRINT 3003, XEDRL, YEDRL, ZEDRL
 3003 FORMAT (1H , 3F10.4, 13X, 'XEDRLM YEDRL, ZEDRL')
      XEDRTL = XEDRL
      YEDRTL = YEDRL
      ZEDRTL = ZEDRL - ZRL
      IF(NOPT.EQ.1)PRINT 2303, XEDRTL, YEDRTL, ZEDRTL
 2303 FORMAT (1H , 3F10.4,  13X, 'XEDRTL, YEDRTK, ZEDRTL')
C         TRANSLATE TO LOCAL COORDINATE SYSTEM WITH ORIGIN AT SURFACE
      XRTL = XRL
      YRTL = YRL
      ZRTL =-ZRL
      XEDP = XRTL + XEDRL
      YEDP = YRTL + YEDRL
      ZEDP = ZRTL + ZEDRL
      RED = SQRT(XEDP*XEDP+YEDP*YEDP+ZEDP*ZEDP)
 2302 FORMAT (1H , 3F10.4,  13X, 'XRTL, YRTL, ZRTL')
      IF(NOPT.EQ.1)PRINT 2302,  XRTL, YRTL, ZRTL
C                                 EARTHS SURFACE AT A SPECIFIED ALTITUDE
C       POSITION OF OFFSET NORTH DIPOLE  IN LOCAL COORDINATE SYSTEM
      XODNPR=   XODNP*CGCROT*CGCT +  YODNP*SGCROT*CGCT -   ZODNP*SGCT
      YODNPR=  -XODNP*SGCROT      +  YODNP*CGCROT
      ZODNPR=   XODNP*CGCROT*SGCT +  YODNP*SGCROT*SGCT +   ZODNP*CGCT
      XODNPT = XODNPR
      YODNPT = YODNPR
      ZODNPT = ZODNPR - ZRL
      IF(NOPT.EQ.1)PRINT 1103, XODNPT, YODNPT, ZODNPT
 1103 FORMAT(1H ,10X,'XODNPT ='F10.4,2X,'YODNPT ='F10.4,2X,'ZODNPT ='
     1 F10.4,3X, 'OFFSET N DIPOLE IN LOCAL COORDINATES')
      ROTM =  ATAN2(YODNPT, XODNPT) + PI
C                 FIND ANGLE FROM GEOGRAPHIC NORTH
C                 NEGATIVE - ROTATION FROM GEOGRAPHIC NP CLOCKWISE
C                 POSITIVE - ROTATION FROM GEOGRAPHIC NP CCW
      SROTM = SIN(ROTM)
      CROTM = COS(ROTM)
      ROTMD = ROTM*RAD
 2327 FORMAT (1H ,  F15.5, 3X, ' ROTM IN DEGREES MEASURED    CCW SO -X
     1 WILL POINT TOWARD OFFSET NORTH DIPOLE AXIS')
      IF(NOPT.EQ.1)PRINT 2327, ROTMD
C         FIND COMPONENTS OF UNIT VECTOR AT ARBITARY AZIMUTH AND ZENITH
      PLAZ = -AZ/RAD + PI
      TLZE = ZE/RAD
      SPLAZ = SIN(PLAZ)
      CPLAZ = COS(PLAZ)
      STLZE = SIN(TLZE)
      CTLZE = COS(TLZE)
```

```
      XLD = STLZE*CPLAZ
      YLD = STLZE*SPLAZ
      ZLD = CTLZE
      IF(NOPT.EQ.1)PRINT 2005, XLD, YLD, ZLD, AZ, ZE
 2005 FORMAT (1H , 5F10.4, 3X, 'UNIT VECTOR COMPOENTS AT AZ & ZE')
C
C        FIND COMPONENTS OF UNIT VECTOR IN DIPOLE RADIAL COORDINATES
C
C       ROTATE AROUND Y AXIS SO -Z AXIS PASSES THROUGH XED, 0, ZED
C       NEW VECTOR IS VA = ZRTL + ZEDRTK + XEDRTL
C       ANGLE BETWEEN VECTOR FROM POINT LOCAL ORIGIN TO GEOCENTER
C        AND VECTOR FROM POINT LOCAL ORIGIN TO XED, 0, ZED
C***********************************************************************
C      JIM LANGWORTHY'S FIX
C***********************************************************************
       CA=DPEC(XEDRTL,ZEDRTL,ZRTL)
C      CA = ZRTL*ZEDRTL/(ABS(ZRTL)*SQRT(ZEDRTL*ZEDRTL + XEDRTL*XEDRTL))
      A = ACOS(CA)
      IF(XEDRTL.GT.0.0)   A = -A
      SA = SIN(A)
      ADEG = A*RAD
      IF(NOPT.EQ.1)PRINT 1000, CA, A, SA, ADEG
      XLP = XLD*CA +     ZLD*SA
      YLP =         YLD
      ZLP =-XLD*SA +     ZLD*CA
      IF(NOPT.EQ.1)PRINT 5001, XLP, YLP, ZLP
 5001 FORMAT(1H , 3F10.4, 13X,'XLP,  YLP,  ZLP  ')
C       ROTATE AROUND X PRIME AXIS SO -Z PASSES THROUGH XED, YED, ZED
       CB=DPEC(YEDRTL,ZEDRTL,ZRTL)
C      CB = ZRTL*ZEDRTL/(ABS(ZRTL)*SQRT(ZEDRTL*ZEDRTL + YEDRTL*YEDRTL))
      B = ACOS(CB)
      IF(YEDRTL.GT.0.0)   B = -B
      SB = SIN(B)
      BDEG = B*RAD
      IF(NOPT.EQ.1)PRINT 1000, CB, B, SB, BDEG
      XLPP = XLP
      YLPP = YLP*CB + ZLP*SB
      ZLPP =-YLP*SB + ZLP*CB
      IF(NOPT.EQ.1)PRINT 5002, XLPP, YLPP, ZLPP
 5002 FORMAT(1H , 3F10.4, 13X,'XLPP, YLPP, ZLPP ')
C       ROTATE AROUND ZPP AXIS SO -X AXIS PASSES THROUGH NORTH
C                                          OFFSET DIPOLE AXIS
      ZLDM = ZLPP
      XLDM = XLPP*CROTM + YLPP*SROTM
      XLDM = XLPP*CROTM  -YLPP*SROTM
      YLDM =-XLPP*SROTM + YLPP*CROTM
      YLDM = XLPP*SROTM + YLPP*CROTM
      IF(NOPT.EQ.1)PRINT 1101, XLD, YLD, ZLD, XLDM, YLDM, ZLDM
 1101 FORMAT (1H , 'UNIT VECTOR IN LOCAL COORDINATES ', 3F8.5,5X,
     1 'UNIT VECTOR IN LOCAL MAGNETIC COORDINATES', 3F10.5)
C        FIND AZUMITH ANGLE OF UNIT VECTOR IN LOCAL DIPOLAR RADIAL COOR
      IF((ABS(YLDM).GT.SMALL).OR.(ABS(XLDM).GT.SMALL)) GO TO 1102
      PAZM=0.0
      GO TO 1104
 1102 PAZM = ATAN2(YLDM,XLDM)
 1104 AZM = (PI - PAZM)*RAD
      IF(AZM.GT.360.0)  AZM = AZM - 360.0
      ZEM = ACOS(ZLDM)*RAD
C        FIND GAMMA ANGLE
      GAMMA = ACOS(YLDM)*RAD
```

```
C       TRANSFORM TO OFFSET DIPOLE COORDINATES
        XED1=XGC-XEDFGC
        YED1=YGC-YEDFGC
        ZED1=ZGC-ZEDFGC
C       FIND THE Z COORDINATE IN OFFSET DIPOLE COORDINATES
        ZED2=XED1*CPST-YED1*SPST+ZED1*CT
C       FIND THE GEOMAGNETIC LATITUDE
        EDLAT=RAD*(PIO2-ACOS(ZED2/RED))
        COSLDA=COS(EDLAT/RAD)
        STORMER=60.*COSLDA**4./
     &          (RED*RED*(1.+SQRT(1.-COSLDA**3.*YLDM))**2)
       RETURN
       END


C-----------------------------------------------------------------------

      REAL FUNCTION DPEC(U,ZEDRTL,ZRTL)

      IMPLICIT NONE
      REAL U,ZEDRTL,ZRTL

C-----------------------------------------------------------------------

      DPEC=SIGN(1./SNGL(DSQRT(1D0+DBLE((U/ZEDRTL)**2))),ZRTL*ZEDRTL)

      RETURN
      END



C-----------------------------------------------------------------------


       SUBROUTINE GridApproxLval(Cgrid,XLbounds,ILbins,Grid80Lval,
     &               UseLapprox)
       IMPLICIT NONE

       REAL XLinfinite,RatioCheck,Grid80Lval,Cgrid
       INTEGER ILmax,NLvals,L,ILbins
       PARAMETER (XLinfinite=1.0E+06,NLvals=10,RatioCheck=1.2)

       REAL XLbounds(NLvals)
       LOGICAL UseLapprox
C-----------------------------------------------------------------------

        Grid80Lval=XLinfinite
        IF (Cgrid .GT. 0.) Grid80Lval=SQRT(14.5/Cgrid)

        ILmax = ILbins

        UseLapprox=.FALSE.

        IF (Grid80Lval .GT. RatioCheck*XLbounds(ILmax)) THEN
          UseLapprox=.TRUE.
        ELSE
          DO L=2,ILmax
           IF ( (Grid80Lval .GT. RatioCheck*XLbounds(L-1)) .AND.
     &         (Grid80Lval .LT. XLbounds(L)/RatioCheck) ) THEN
             UseLapprox=.TRUE.
```

```
    ENDIF
   ENDDO
ENDIF

RETURN
END
```

```
      SUBROUTINE GET_BLCOORDS(YEAR,LATI,LONGI,HEIGHT,XL,BB0,ICODE)

C     Subroutine -- adapted from BILCAL by AJT -- for calculating
C     geomagnetic coordinates B/B0 (=BB0) and McIlwain L (=XL)
C     12/9/92
C
C     Modified 11-17-97:  add IMPLICIT NONE & variable-type declarations
C
C     Inputs:
C     YEAR = year (eg., 1987.63) for field initialization, etc.
C     LATI,LONGI = geodetic latitude and (east) longitude (degrees)
C     HEIGHT     = geodetic altitude (km above sea level)
C     Outputs:
C     XL = McIlwain L parameter
C     BB0 = B/B0
C     ICODE = return code: 1=OK; 3=approx result;
C                          2=one of the conjugate mirror points is
C                            unphysical.

      IMPLICIT NONE

      REAL YEAR,LATI,LONGI,HEIGHT,XL,BB0
      INTEGER ICODE
      LOGICAL VAL
      REAL DIMO,BNORTH,BEAST,BDOWN, BABS, BAB1,BEQU,BDEL,BEQ,RR0

C     Initialize field coefficients (if needed), get dipole moment
      CALL FELDCOF(YEAR,DIMO)
C     Get local field strength (BABS)
      CALL FELDG(LATI,LONGI,HEIGHT,BNORTH,BEAST,BDOWN,BABS)
C     Calculate McIlwain L and set ICODE flag.
      CALL SHELLG(LATI,LONGI,HEIGHT,DIMO,XL,ICODE,BAB1)
      IF(IABS(ICODE).GT.9) ICODE=2
C     Calculate B/B0
      BEQU=DIMO/(XL*XL*XL)
      IF(ICODE.EQ.1) THEN
              BDEL=1.E-3
              CALL FINDB0(0.05,BDEL,VAL,BEQ,RR0)
              IF(VAL) BEQU=BEQ
      ENDIF
      BB0=BABS/BEQU
      IF(BB0.GT.9999.999) BB0=9999.999
C     Done
      RETURN
      END
```

```
      SUBROUTINE GET_CHECK_CONTROL(FILE_CHECK)
C
C     Sets logical to direct various file-checking functions in the
C     CREME96 software (CHECK_FILE, CHECK_NAME_CONFLICT,CHECK_OUTPUT_FILE)
C
C     Recommended usage: logical FILE_CHECK=.true.
C                        for VAX & PC stand-alone versions of the code
C
C                        logical FILE_CHECK=.false.
C                        for the PC/WWW version.
C
      IMPLICIT NONE
      LOGICAL FILE_CHECK

      FILE_CHECK=.true.

      RETURN
      END
```

```
      REAL FUNCTION GET_CREME96_FLUX(IZ,EN,YEAR,IMODE,ITRANS)
C
C     Returns particle flux from CREME96 particle environment model.
      IMPLICIT NONE
      INTEGER*4 IZ,IMODE,ITRANS
      REAL*4 EN,YEAR,FLUX
      REAL*4 CRF96,CUT96,GET_TRAPPED_PROTONS,GET_TRAPPED_IONS

      GET_CREME96_FLUX=0.0

      IF (EN.LT.1.0) RETURN
      IF (IZ.LT.1 .or. IZ.GT.92) RETURN

      IF (ITRANS.EQ.0) THEN
C         Fluxes outside of the magnetosphere
          FLUX=CRF96(IZ,EN,YEAR,IMODE)
      ELSEIF (ITRANS.EQ.1) THEN
C         Non-trapped fluxes inside the magnetosphre
          FLUX=CUT96(IZ,EN,YEAR,IMODE)
      ELSEIF (ITRANS.EQ.2) THEN
C         Non-trapped & Trapped fluxes inside the magnetosphere
          FLUX=CUT96(IZ,EN,YEAR,IMODE)
C         Function names modified 12-10-97 by AJT
          IF (IZ.EQ.1) FLUX=FLUX+GET_TRAPPED_PROTONS(EN)
          IF (IZ.GT.1) FLUX=FLUX+GET_TRAPPED_IONS(IZ,EN)
      ENDIF

      GET_CREME96_FLUX=FLUX

      RETURN
      END
```

```
      SUBROUTINE GET_CREME96_FLUX_MODEL(INFILE,MODEL_TYPE)
C
C     Decodes header of CREME96 flux file to determine type of flux model.
C     Information is used in converting solar particle results from average
C     rates to event-accumulated numbers.
C
      IMPLICIT NONE
      CHARACTER*80 INFILE,ILINE
      CHARACTER*3 SUFFIX,IMODEL
      INTEGER*4 MODEL_TYPE,ILONG,ILONG1,ILONG2,
     &          IVER,J,NHEADER,STAT,CREME96_OPEN

      MODEL_TYPE=0

      ILONG=INDEX(inFILE,'.')
      SUFFIX=INFILE(ILONG+1:ILONG+3)
      CALL CAPITALIZE_STRING(SUFFIX,3)

      IF (SUFFIX.ne.'FLX' .and. SUFFIX.ne.'TFX' .and.
     &    SUFFIX.ne.'LET' .and. SUFFIX.ne.'DLT') RETURN

C
C     Now open file and decode header:
C
      CALL CHECK_CREME96_VERSION(INFILE,IVER)

      IF (IVER.GE.102) THEN
          stat = creme96_open(infile,'user',10,'old')
          IF (STAT.EQ.0) THEN
          READ(10,*) NHEADER
          DO J=1,NHEADER
             READ(10,110) ILINE
 110         FORMAT(A80)
             IMODEL='   '
             ILONG1=INDEX(ILINE,'%IMODE =')
             IF (ILONG1.NE.0) IMODEL=ILINE(ILONG1+8:ILONG1+10)
             ILONG2=INDEX(ILINE,'%IMODE   =')
             IF (ILONG2.NE.0) IMODEL=ILINE(ILONG2+10:ILONG2+12)
             IF (IMODEL.NE.'   ') THEN
C                 WRITE(6,9999) IMODEL
C 9999            FORMAT(' IMODEL=',A3,'====')
                DECODE(3,100,IMODEL) MODEL_TYPE
 100            FORMAT(I3)
                RETURN
             ENDIF
           ENDDO
           ENDIF
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE GET_CREME96_VERSION(IVER)
C
C     Sets version number of CREME96 software, for record keeping purposes:
C
      IMPLICIT NONE
      INTEGER*4 IVER

C     Modified 7/29/96: Version 1.01

C     Modified 8/19/96 Version 1.02; more extensive output file headers

C     Modified 9/14/96 Version 1.03: default energy limits;
C                                    energy limits in LET calculations
C                                    extended tables for Z > 28 added.
C     Modified 9/25/96 Version 1.04: ACR charge-state distributions added
      IVER=104

      RETURN
      END
```

```
      REAL FUNCTION GET_GTF(RIGIDITY)
C
C     Evaluates orbit-averaged geomagnetic transmission function
C     (previously calculated by GEOMAG96 and loaded into COMMON/GTFDAT
C     by LOAD_GTF) at rigidity RIGIDITY (in GV).
C
      IMPLICIT NONE
      INTEGER*4 NGTF,IGTF,I,ISAV
      REAL*4 R,GTF,P,RIGIDITY
      PARAMETER (NGTF=1001)
      COMMON/GTFDAT/IGTF,R(NGTF),GTF(NGTF)
C
C     LOOK UP THE TABULATED MAGNETIC RIGIDITY JUST ABOVE RIGID
C
      GET_GTF=0.0
      IF (IGTF.LE.0) RETURN

      P=RIGIDITY

      GET_GTF=GTF(IGTF)
      IF (P.GT.R(IGTF)) RETURN

      GET_GTF=0.0
      IF (P.LT.R(1)) RETURN

 4    DO 2 I=2,IGTF
      IF(P.GT.R(I)) GOTO 2
      ISAV=I
      GOTO 3
 2    CONTINUE
C
C     INTERPOLATE THE TRANSMISSION FACTOR (AVERAGED FOR THE ORBIT).
C
 3    GET_GTF=GTF(ISAV-1)+
     *  (GTF(ISAV)-GTF(ISAV-1))*(P-R(ISAV-1))/(R(ISAV)-R(ISAV-1))


      RETURN
      END
```

```
      SUBROUTINE GET_HI_XS_INPUTS(DEVICE_LABEL,XM,YM,ZM,FUNNELM,NBITS,
     &                            IPARAM,PARAMS,XSECT_FILE)
C
C
C     Interactive dialogue to get necessary cross-sectioninput parameters
C     for heavy-ion upsets:
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C-----------------------------------------------------------------------
C
      IMPLICIT NONE
      CHARACTER*80 XSECT_FILE
      CHARACTER*40 DEVICE_LABEL
      REAL*4 XM,YM,ZM,FUNNELM,PARAMS,NBITS
      INTEGER*4 IPARAM
      DIMENSION PARAMS(4)
      INTEGER*4 IFILETYPE,IACCEPT
      INTEGER*4 IERR
      DATA IERR/0/

      WRITE(6,1210)
 1210 FORMAT(1x,' *** NOTE ***:  At any point in the following'
     &          ' dialogue, you can go back and'
     &     /,1x,' change a parameter by entering -1 for the',
     &          ' presently requested information.',
     &     /,1x,' Repeated -1 values can be used to scroll back',
     &          ' to (almost) anywhere in the input menu.')

  105 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1215)
 1215 FORMAT(/,1x,' Enter device label and/or comments',
     &        ' (40 characters max) for record-keeping:')
      READ(*,1218,ERR=105,IOSTAT=IERR) DEVICE_LABEL
 1218 FORMAT(A40)

      IF (DEVICE_LABEL(1:2).EQ.'-1') RETURN
      WRITE(6,1219) DEVICE_LABEL
 1219 FORMAT(1x,' Device/Comment: ',A40)

 1185 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1200)
 1200 FORMAT(/,' This program calculates the heavy-ion SEU rate',
     &          ' using the RPP method.',
     &        /,' Enter the dimensions of the',
     &          ' bit sensitive volume: (X,Y,Z; in microns)')

      READ(*,*,ERR=1185,IOSTAT=IERR) XM,YM,ZM
      IF (XM.LE.-1.0 .or. YM.LE.-1.0 .or. ZM.LE.-1.0) GOTO 105

      WRITE(6,1220) XM,YM,ZM
 1220 FORMAT(' Sensitive volume dimensions = ',
     &          F8.2,' x ',F8.2,' x ',F8.2,' microns')
```

```
1195  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,2200)
2200  FORMAT(' Enter funnel length (microns): ')
      READ(*,*,ERR=1195,IOSTAT=IERR) FUNNELM
      IF (FUNNELM.LE.-1.0) GOTO 1185
      WRITE(6,2220) FUNNELM
2220  FORMAT(' Funnel length = ',F8.2,' microns.')

1225  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1300)
1300  FORMAT(//,' This code supports several methods for specifying',
     &          ' the SEU cross-section:',
     &        /,' METHOD = 0: a file containing a two-column table,'
     &        /,' METHOD = 1: Bendel 1-parameter fit ',
     &        /,' METHOD = 2: Bendel 2-parameter fit ',
     &        /,' METHOD = 3: NOT CURRENTLY USED',
     &        /,' METHOD = 4: Weibull fit ',
     &        /,' METHOD = 5: Critical charge (in pC)',
     &       //,' Specify METHOD (0,1,2,4, or 5): ')

      READ(*,*,ERR=1225,IOSTAT=IERR) IPARAM
      IF (IPARAM.LE.-1) GOTO 1195

      IF (IPARAM.EQ.0) THEN
1395     CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,1400)
1400     FORMAT(' SEU cross-section from input table file:',
     &          /,' This table must have a two-column format with :',
     &   /,11x,   ' column 1 containing LET (in MeV-cm2/milligram)',
     &   /,7x,' and column 2 containing SEU cross-section',
     &        ' (in sq. microns/bit)',
     &   /,7x,' and be ordered according to increasing LET.',
     &   /,' The file containing the table must already exist in',
     &     ' your current USER area and ',
     &   /,' be called something.XSD (ie., have XSD for the extension).',
     &   /,' Enter name of the cross-section file: ')

      READ(*,1,ERR=1395,IOSTAT=IERR) XSECT_FILE
1     FORMAT(A80)
      IF (XSECT_FILE(1:2).EQ.'-1') GOTO 1225
      WRITE(6,1410) XSECT_FILE
1410  FORMAT(1x,' Input Heavy-Ion Cross-Section File = ',/,1x,A80)
      IFILETYPE=8
      CALL CHECK_FILE(IFILETYPE,XSECT_FILE,IACCEPT)
      IF (IACCEPT.NE.0) GOTO 1395

      ELSEIF (IPARAM.EQ.1) THEN
1495     CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,1500)
1500     FORMAT(' Bendel 1-parameter fit to the cross-section: ',
     &          /,'   NOTE: Your fit parameters must specify heavy-ion',
     &          /,'   SEU cross-section (in sq. microns/bit) vs.',
     &            ' LET (in MeV-cm2/milligram): '
     &          /,' Enter Bendel-1 parameter value: ')
         READ(*,*,ERR=1495,IOSTAT=IERR) PARAMS(1)
```

```
         IF (PARAMS(1).LE.-1.) GOTO 1225
         WRITE(6,1510) PARAMS(1)
1510     FORMAT(' Bendel-1 parameter = ',E13.6)

      ELSEIF (IPARAM.EQ.2) THEN
1595     CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,1600)
1600     FORMAT(' Bendel 2-parameter fit to the cross-section: ',
     &        /,'   NOTE: Your fit parameters must specify heavy-ion',
     &        /,'   SEU cross-section (in sq. microns/bit) vs.',
     &          ' LET (in MeV-cm2/milligram): '
     &        /,' Enter Bendel A & B parameter values: ')
         READ(*,*,ERR=1595,IOSTAT=IERR) PARAMS(1),PARAMS(2)
         IF (PARAMS(1).LE.-1. .or. PARAMS(2).LE.-1.) GOTO 1225
         WRITE(6,1610) PARAMS(1),PARAMS(2)
1610     FORMAT(' Bendel parameters A,B = ',2E13.6)

      ELSEIF (IPARAM.EQ.3.or.IPARAM.LT.0 .or. IPARAM.GT.5) THEN
1695     CONTINUE
         WRITE(6,1700)
1700     FORMAT(' ILLEGAL CROSS-SECTION SPECIFICATION CODE. ',
     &        /,' Please try again.')
         GOTO 1225

      ELSEIF (IPARAM.EQ.4) THEN
1795     CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,1800)
1800     FORMAT(' Weibull fit to the cross-section: ',
     &        /,'   NOTE: Your fit parameters must specify heavy-ion'
     &        /,'   SEU cross-section (in sq. microns/bit) vs.',
     &          ' LET (in MeV-cm2/milligram): ')
         WRITE(6,1810)
1810     FORMAT(' Enter ONSET parameter (in MeV-cm2/milligram): ')
         READ(*,*,ERR=1795,IOSTAT=IERR) PARAMS(1)
         IF (PARAMS(1).LE.-1.) GOTO 1225
1815     CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,1820)
1820     FORMAT(' Enter WIDTH parameter (in MeV-cm2/milligram): ')
         READ(*,*,ERR=1815,IOSTAT=IERR) PARAMS(2)
         IF (PARAMS(2).LE.-1.) GOTO 1795
1825     CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,1830)
1830     FORMAT(' Enter POWER parameter (dimensionless exponent): ')
         READ(*,*,ERR=1825,IOSTAT=IERR) PARAMS(3)
         IF (PARAMS(3).LE.-1.) GOTO 1815
1835     CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,1840)
         IF (XM.GT.0.0 .and. YM.GT.0.0) WRITE(6,1841)
1840     FORMAT(' Enter cross-section plateau value',
     &          ' (in sq. microns/bit).')
1841     FORMAT(' (If 0, calculation will use surface area (xy) of',
     &          ' the RPP sensitive volume.)')
         READ(*,*,ERR=1835,IOSTAT=IERR) PARAMS(4)
         IF (PARAMS(4).LE.-1.) GOTO 1825
         IF (PARAMS(4).LE.0.) PARAMS(4)=XM*YM
```

```
          WRITE(6,1850) PARAMS(1),PARAMS(2),PARAMS(3),PARAMS(4)
1850      FORMAT(' Weibull fit parameters: ',
     &          /,5x,' ONSET   = ',F9.3,' MeV-cm2/milligram',
     &          /,5x,' WIDTH   = ',F9.3,' MeV-cm2/milligram',
     &          /,5x,' POWER   = ',F9.3,' (dimensionless)',
     &          /,5x,' PLATEAU = ',F9.3,' square microns/bit')

      ELSEIF (IPARAM.EQ.5) THEN
1895      CONTINUE
          WRITE(6,1900)
1900      FORMAT(' Cross-section given as step function in critical',
     &             ' charge.',
     &    /,' NOTE: in general this method does NOT'
     &      ' give accurate results for space',
     &    /,' applications, but it may be useful for'
     *      ' order-of-magnitude estimates by chip',/,' designers.')
1905      CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,1910)
1910      FORMAT(' Enter critical charge (in picocoloubs): ')
          READ(*,*,ERR=1905,IOSTAT=IERR) PARAMS(1)
          IF (PARAMS(1).LE.-1.) GOTO 1225
1915      CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,1920)
          IF (XM.GT.0.0 .and. YM.GT.0.0) WRITE(6,1841)
1920      FORMAT(' Enter cross-section (in square microns/bit):')
          READ(*,*,ERR=1915,IOSTAT=IERR) PARAMS(2)
          IF (PARAMS(2).LE.-1.) GOTO 1905
          IF (PARAMS(2).LE.0.) PARAMS(2)=XM*YM
          WRITE(6,1930) PARAMS(1),PARAMS(2)
1930      FORMAT(' Critical charge = ',E13.5,' picocoloumbs',
     &         /,' Cross-Section   = ',E13.5,' square microns/bit')

       ENDIF

1995   CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,2000)
2000   FORMAT(' Finally, specify number of bits per device: ')
       READ(*,*,ERR=1995,IOSTAT=IERR) NBITS
       IF (NBITS.EQ.-1) THEN
          IF (IPARAM.EQ.1) GOTO 1495
          IF (IPARAM.EQ.2) GOTO 1595
          IF (IPARAM.EQ.3) GOTO 1695
          IF (IPARAM.EQ.4) GOTO 1835
          IF (IPARAM.EQ.5) GOTO 1915
       ENDIF

       WRITE(6,2010) NBITS
2010   FORMAT(1x,E13.5,' bits per device.')

       RETURN
       END
```

```
      SUBROUTINE GET_PROTON_XS_INPUTS(DEVICE_LABEL,NBITS,
     &                                IPARAM,PARAMS,XSECT_FILE)
C
C
C     Generates interactive dialogue to get necessary input parameters
C     for proton-induced SEU rate:
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C     Last update:   20 August 1996
C
C-----------------------------------------------------------------------
C
      IMPLICIT NONE
      CHARACTER*80 XSECT_FILE
      CHARACTER*40 DEVICE_LABEL
      REAL*4 PARAMS,NBITS
      INTEGER*4 IPARAM
      DIMENSION PARAMS(4)
      INTEGER*4 IFILETYPE,IACCEPT,IERR
      DATA IERR/0/

      WRITE(6,1210)
 1210 FORMAT(1x,' *** NOTE ***:  At any point in the following'
     &           ' dialogue, you can go back and'
     &     /,1x,' change a parameter by entering -1 for the',
     &           ' presently requested information.',
     &     /,1x,' Repeated -1 values can be used to scroll back',
     &           ' to (almost) anywhere in the input menu.')


  105 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1215)
 1215 FORMAT(/,1x,' Enter device label and/or comments',
     &        ' (40 characters max) for record-keeping:')
      READ(*,1218,ERR=105,IOSTAT=IERR) DEVICE_LABEL
 1218 FORMAT(A40)

      IF (DEVICE_LABEL(1:2).EQ.'-1') RETURN
      WRITE(6,1219) DEVICE_LABEL
 1219 FORMAT(1x,' Device/Comment: ',A40)


 1295 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1300)
 1300 FORMAT( /,' This code supports several methods for specifying',
     &            ' the SEU cross-section:',
     &         /,' METHOD = 0: a file containing a two-column table,'
     &         /,' METHOD = 1: Bendel 1-parameter fit ',
     &         /,' METHOD = 2: Bendel 2-parameter fit ',
     &         /,' METHOD = 3: NOT CURRENTLY USED',
     &         /,' METHOD = 4: Weibull fit ',
     &        //,' Specify METHOD (0,1,2, or 4): ')
```

```
          READ(*,*,ERR=1295,IOSTAT=IERR) IPARAM
          IF (IPARAM.EQ.-1) GOTO 105
          IF (IPARAM.EQ.0) THEN
 1395        CONTINUE
             CALL RETRY_INPUT(IERR)
             WRITE(6,1400)

 1400        FORMAT(' SEU cross-section from input table file:',
     &             /,' This table must have a two-column format with :',
     &   /,11x,    ' column 1 containing proton energy (in MeV)',
     &   /,7x,' and column 2 containing SEU cross-section',
     &        ' (in 10**-12 cm2/bit)',
     &   /,7x,' and be ordered according to increasing proton energy.',
     &   /,' The file containing the table must already exist in',
     &     ' your current USER area and ',
     &   /,' be called something.XSD (ie., have XSD for the extension).',
     &   /,' Enter name of the cross-section file: ')

          READ(*,1,ERR=1395,IOSTAT=IERR) XSECT_FILE
 1        FORMAT(A80)
          IF (XSECT_FILE(1:2).EQ.'-1') GOTO 1295
          WRITE(6,1410) XSECT_FILE
 1410     FORMAT(1x,' Input Proton Cross-Section File = ',/,1x,A80)
          IFILETYPE=8
          CALL CHECK_FILE(IFILETYPE,XSECT_FILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 1395

          ELSEIF (IPARAM.EQ.1) THEN
 1495        CONTINUE
             CALL RETRY_INPUT(IERR)
             WRITE(6,1500)
 1500        FORMAT(' Bendel 1-parameter fit to the cross-section: ',
     &             /,'   NOTE: Your fit parameters must specify proton',
     &             /,'   SEU cross-section (in 10**-12 cm2/bit) vs.',
     &               ' proton energy (in MeV): '
     &             /,' Enter Bendel-1 parameter value: ')
             READ(*,*,ERR=1495,IOSTAT=IERR) PARAMS(1)
             IF (PARAMS(1).LE.-1) GOTO 1295
             WRITE(6,1510) PARAMS(1)
 1510        FORMAT(' Bendel-1 parameter = ',E13.6)

          ELSEIF (IPARAM.EQ.2) THEN
 1595        CONTINUE
             CALL RETRY_INPUT(IERR)
             WRITE(6,1600)
 1600        FORMAT(' Bendel 2-parameter fit to the cross-section: ',
     &       /,'   NOTE: Your fit parameters must specify the proton',
     &       /,'   SEU cross-section (in 10**-12 cm2/bit) vs.',
     &         ' proton energy (in MeV): '
     &       /,' Enter Bendel A & B parameter values: ')
             READ(*,*,ERR=1595,IOSTAT=IERR) PARAMS(1),PARAMS(2)
             IF (PARAMS(1).LE.-1. .or. PARAMS(2).LE.-1.) GOTO 1295
             WRITE(6,1610) PARAMS(1),PARAMS(2)
 1610        FORMAT(' Bendel parameters A,B = ',2E13.6)

          ELSEIF (IPARAM.EQ.3.or.IPARAM.LT.0 .or. IPARAM.GT.4) THEN
 1695        CONTINUE
             WRITE(6,1700)
 1700        FORMAT(' ILLEGAL CROSS-SECTION SPECIFICATION CODE. ',
     &             /,' Please try again.')
```

```
          GOTO 1295

        ELSEIF (IPARAM.EQ.4) THEN
1795      CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,1800)
1800      FORMAT(' Weibull fit to the cross-section: ',
     &    /,'   NOTE: Your fit parameters must specify the proton'
     &    /,'   SEU cross-section (in 10**-12 cm2/bit) vs.',
     &      ' proton energy (in MeV): ')
          WRITE(6,1810)
1810      FORMAT(' Enter ONSET parameter (in MeV): ')
          READ(*,*,ERR=1795,IOSTAT=IERR) PARAMS(1)
          IF (PARAMS(1).LE.-1.) GOTO 1295
1815      CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,1820)
1820      FORMAT(' Enter WIDTH parameter (in MeV): ')
          READ(*,*,ERR=1815,IOSTAT=IERR) PARAMS(2)
          IF (PARAMS(2).LE.-1.) GOTO 1795
1825      CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,1830)
1830      FORMAT(' Enter POWER parameter (dimensionless exponent): ')
          READ(*,*,ERR=1825,IOSTAT=IERR) PARAMS(3)
          IF (PARAMS(3).LE.-1.) GOTO 1815
1835      CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,1840)
1840      FORMAT(' Enter cross-section plateau value',
     &              ' (in 10**-12 cm2/bit):')
          READ(*,*,ERR=1835,IOSTAT=IERR) PARAMS(4)
          IF (PARAMS(4).LE.-1.) GOTO 1825
          WRITE(6,1850) PARAMS(1),PARAMS(2),PARAMS(3),PARAMS(4)
1850      FORMAT(' Weibull fit parameters: ',
     &          /,5x,' ONSET   = ',F9.3,' MeV',
     &          /,5x,' WIDTH   = ',F9.3,' MeV',
     &          /,5x,' POWER   = ',F9.3,' (dimensionless)',
     &          /,5x,' PLATEAU = ',F9.3,' x 10**-12 cm2/bit')

        ENDIF

1995    CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(6,2000)
2000    FORMAT(' Finally, specify number of bits per device: ')
        READ(*,*,ERR=1995,IOSTAT=IERR) NBITS
        IF (NBITS.EQ.-1) THEN
          IF (IPARAM.EQ.1) GOTO 1495
          IF (IPARAM.EQ.2) GOTO 1595
          IF (IPARAM.EQ.3) GOTO 1695
          IF (IPARAM.EQ.4) GOTO 1835
        ENDIF
        WRITE(6,2010) NBITS
2010    FORMAT(1x,E13.5,' bits per device.')

        RETURN
        END
```

```
      REAL FUNCTION GET_TRAPPED_IONS(IZ,EN)
C
C     Returns orbit-averaged flux of trapped ion IZ at energy EN
C     -->  NOT INCLUDED IN CREME96
C
      GET_TRAPPED_IONS=0.0
      RETURN
      END
```

```
      REAL FUNCTION GET_TRAPPED_PROTONS(EN)
C
C     Returns orbit-averaged trapped proton flux (in protons/m2-s-sr-MeV)
C     at energy EN (in MeV) by interpolating value from previously stored
C     array.
C
C     Formerly called "TRAPPED_PROTONS".  Renamed by AJT 12-9-97, to
C     remove name conflict with new PRB/SAIC routines on trapped protons.
C
C

      IMPLICIT NONE
      INTEGER*4 MAXSPEC,ITRPSPEC,I
      REAL*4 EN,ENTRP,FLUXTRP
      PARAMETER (MAXSPEC=5000)
      COMMON/TRPDAT/ITRPSPEC,ENTRP(MAXSPEC),FLUXTRP(MAXSPEC)
      REAL*4 X1,Y1,X2,Y2,X3,Y3,SLOPE

      GET_TRAPPED_PROTONS=0.

      IF (EN.LT.ENTRP(1) .or. EN.GT.ENTRP(ITRPSPEC)) RETURN

      DO 100 I=2,ITRPSPEC
             IF (EN.LE.ENTRP(I)) THEN
                 IF (FLUXTRP(I).GT.0. .and. FLUXTRP(I-1).GT.0.) THEN
                     X1=ALOG(ENTRP(I-1))
                     Y1=ALOG(FLUXTRP(I-1))
                     X2=ALOG(ENTRP(I))
                     Y2=ALOG(FLUXTRP(I))
                     SLOPE=(Y2-Y1)/(X2-X1)
                     X3=ALOG(EN)
                     Y3=SLOPE*(X3-X1)+Y1
                     GET_TRAPPED_PROTONS=EXP(Y3)
                     GOTO 150
                 ELSE
                     GET_TRAPPED_PROTONS=0.0
                 ENDIF
             ENDIF
100       CONTINUE
150       CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GET_UPSET(XM,YM,ZM,FUNNELM,QCRIT,
     &                      NVAL,LVAL,FVAL,UPSET)
C
C     Subroutine for evaluating SEUs as a function of critical
C     charge.  Derived from the UPSET program in the original CREME
C     software.
C
C     Inputs:  XM,YM,ZM =  bit dimensions x,y,z (in microns)
C              FUNNELM   =  funnel length (in microns)
C              Qcrit     =  (critical charge) in picocoulombs
C              LVAL,FVAL = arrays containing integral LET spectrum
C                          in particles/m2-sr-s
C                          vs. LET in MeV-cm2/gm
C                          in NVAL points
C              Output: UPSET = SEUs/bit/s
C
C     Modified by AJT 10-24-96: increases density of sampling in
C     LET spectrum around values corresponding to peaks in the differential
C     pathlength distribution.  This removes a previously noted problem
C     in the old GET_UPSET routine, in which the SEU rate was not a
C     strictly non-increasing function of increasing critical charge.
C
C     Modified by AJT 10-30-96: funnels added.
C
C     Modified by AJT 11-07-96: changed to trapezoidal rule integration,
C                               as suggested by Ed Petersen.
C
C-----------------------------------------------------------------------
C     BANNER FROM CREME 1981 VERSION:
C
C     THIS PROGRAM COMPUTES THE UPSET RATE DUE TO THE
C     DIRECT IONIZATION OF INDIVIDUAL PARTICLES.  IT ASSUMES
C     THAT FOR EACH BIT THERE EXISTS A SENSITIVE VOLUME.
C     IF AN AMOUNT OF ELECTRICAL CHARGE (>QCRIT) IS CREATED
C     WITHIN THIS VOLUME BY THE IONIZING PARTICLE, THEN AN
C     UPSET WILL OCCUR.  THE SENSITIVE VOLUME IS IDEALIZED AS
C     A PARALLELEPIPED WITH DIMENSIONS X, Y, AND Z.
C     THE OUTPUT IS GIVEN IN UPSETS/(BIT*SECOND) AND UPSETS/(BIT*DAY).
C
C     THIS CALCULATION USES THE METHOD DESCRIBED IN "THE VARIABILITY
C     OF SINGLE EVENT UPSET RATES IN THE NATURAL ENVIRONMENT,"
C     JAMES H. ADAMS, JR., IEEE TRANS. ON NUCL. SCI., NS-30, 4475-
C     4480, DEC., 1983.
C     IT IS OPERATIONALLY EQUIVALENT TO ROCKWELL'S CRIER PROGRAM
C     (PICKEL AND BLANDFORD, IEEE TRANS. ON NUCL. SCI. NS-26, DEC.
C     1979, PP.4735-4739) WHEN USED WITH HEINRICH'S LET SPECTRUM
C     (W. HEINRICH, RADIATION EFFECTS, VOL. 34, PP. 143-8, 1977).
C
C     THIS PROGRAM CALLS A SUBPROGRAM, DIFPLD, THAT RETURNS
C     THE DIFFERENTAL PATHLENGTH DISTRIBUTION IN THE SENSTIVE VOLUME
C-----------------------------------------------------------------------

      IMPLICIT NONE
      INTEGER*4 NBINS,NVAL,I,K,N
      REAL*4 XM,YM,ZM,FUNNELM,QCRIT,LVAL,FVAL,UPSET
      REAL*4 L,FLUX,X,Y,Z,DSI,AREA,PMAX,ENERGY,LMIN,SUM,Q
      REAL*4 D,DIFPLD
      REAL*4 FUNNEL
      REAL*4 INTGRND0,INTGRND
      PARAMETER (NBINS=5000)
```

```
      DIMENSION L(NBINS),FLUX(NBINS)
      DIMENSION LVAL(1),FVAL(1)
      DATA DSI/2.321/
      REAL*4 LUSE,FUSE
      DIMENSION LUSE(NBINS),FUSE(NBINS)
      INTEGER*4 M,IENTER
C
C     Load the inputs
      IF (IENTER.EQ.0) THEN
          IENTER=1
          WRITE(6,9999)
 9999     FORMAT(1X,' GET_UPSET revision 11/08/96 active: ')
      ENDIF

      UPSET=0.0
C
      CALL TEMP_STORAGE(NBINS,NVAL,LVAL,FVAL,N,L,FLUX)

C
C     RPP Dimensions:
C
      X=XM
      Y=YM
      Z=ZM
      FUNNEL=FUNNELM
C
C     CONVERT FROM MICROMETERS TO CENTIMETERS.
C
      X=X*.0001
      Y=Y*.0001
      Z=Z*.0001
      FUNNEL=FUNNEL*0.0001
C
C     COMPUTE THE SURFACE AREA OF THE SENSITIVE VOLUME.
C
      AREA=(2.*X*Y+2.*X*Z+2.*Y*Z)
C
C     CONVERT FROM SQUARE CENTIMETERS TO SQUARE METERS.
C
      AREA=AREA*.0001
C
C     CONVERT THE DIMENSIONS OF THE SENSITIVE VOLUME TO G/CM**2.
C
      X=X*DSI
      Y=Y*DSI
      Z=Z*DSI
      FUNNEL=FUNNEL*DSI
C
C     COMPUTE THE MAJOR DIAMETER OF THE SENSITIVE VOLUME.
C
      PMAX=SQRT(X*X+Y*Y+Z*Z)
C
C     COMPUTE THE ENERGY (IN MEV) REQUIRED TO PRODUCE QCRIT(IN PC)
C     HOLE-ELECTRON PAIRS IN SILICON.
C
      ENERGY=22.5*QCRIT
C
C     COMPUTE THE MINIMUM LET THAT CAN PRODUCE AN UPSET.
C
C     Funnel added to this equation 10/30/96 AJT
```

```
C         LMIN=ENERGY/PMAX
          LMIN=ENERGY/(PMAX+FUNNEL)
C
C         Modification AJT 10/24/96: expand sampling around discontinuities
C         in the pathlength density distribution
C
          CALL EXPAND_SAMPLING(NBINS,N,L,FLUX,LMIN,ENERGY,PMAX,X,Y,Z,
     &                               M,LUSE,FUSE)
C
C
C         Now use expanded sampling in numerical integration:
C
C         INTEGRATE FROM LMIN TO THE LARGEST LET IN THE SPECTRUM.
C
          SUM=0.0
          Q=LMIN

C         Note: to uses the trapezoidal rule in the numerical integration,
C         we need to evaluate the integrand (DIFPLD*FLUX/L**2) on a grid
C         which includes the endpoints of the integration.  The lower endpoint
C         is at LMIN, which corresponds to PMAX, the longest possible path
C         through the RPP.  However, DIFPLD=0 at PMAX.  Thus, the integrand
C         also vanishes at LMIN.

          INTGRND0=0.0

          DO 10 I=1,M

          IF (LUSE(I).LT.LMIN) GO TO 10
C
C         Terminate numerical integration when integral flux falls to zero.

          IF (FUSE(I).LE.0.0) GOTO 11
C
C         COMPUTE THE PATHLENGTH CORRESPONDING TO L(I).
C
C         Funnel added to this equation 10/30/96 AJT
C         D=ENERGY/LUSE(I)
          D=(ENERGY/LUSE(I)-FUNNEL)
          IF (D.LT.0.0) D=0.0


C
C         CARRY OUT THE INTEGRAL.

C         Modified to use trapezoidal rule 11/7/96:
COLD      SUM=SUM+(LUSE(I)-Q)*DIFPLD(D,X,Y,Z)*FUSE(I)/(LUSE(I)**2)

          INTGRND=DIFPLD(D,X,Y,Z)*FUSE(I)/(LUSE(I)**2)
          SUM=SUM+0.5*(LUSE(I)-Q)*(INTGRND+INTGRND0)
          INTGRND0=INTGRND

          Q=LUSE(I)

 10       CONTINUE

 11       CONTINUE
C
C         COMPUTE THE ERROR RATE.
C
 12       CONTINUE
```

```
      UPSET=ENERGY*AREA*3.1416*SUM

      RETURN
      END


C-----------------------------------------------------------------------

      SUBROUTINE TEMP_STORAGE(NBINS,NVAL,LVAL,FVAL,N,L,FLUX)

      IMPLICIT NONE
      INTEGER*4 NBINS,NVAL,N,K
      REAL*4 LVAL,FVAL,L,FLUX
      DIMENSION LVAL(1),FVAL(1),L(1),FLUX(1)

C     Copy Integral LET spectrum:
      DO 5 K=1,NVAL
              L(K)=LVAL(K)
              FLUX(K)=FVAL(K)
 5    CONTINUE
      N=NVAL

      IF (N.GT. NBINS) THEN
      WRITE(6,9999) N, NBINS
 9999 FORMAT('@ 10002 ABNORMAL TERMINATION: ',
     &   /,1x,' LET spec array out of bounds in GET_UPSET: ',
     &   /,1x,' N = ',I8,' NBINS = ',I8,
     &   /,1x,' STOP.')
      STOP
      ENDIF
      RETURN
      END

C-----------------------------------------------------------------------

      SUBROUTINE EXPAND_SAMPLING(NBINS,N,L,FLUX,
     &                          LMIN,ENERGY,PMAX,X,Y,Z,
     &                          M,LUSE,FUSE)

      IMPLICIT NONE
      INTEGER*4 NBINS,N,M,TBINS
      REAL*4 L,FLUX,LMIN,ENERGY,PMAX,X,Y,Z,LUSE,FUSE
      DIMENSION L(1),FLUX(1),LUSE(1),FUSE(1)
      PARAMETER (TBINS=5000)
      REAL*4 LTEMP,FTEMP
      DIMENSION LTEMP(TBINS),FTEMP(TBINS)
      INTEGER*4 I,K
      REAL*4 S,STEMP,SDUM,XVAL,YVAL
      REAL*4 SCALE,SMALLEST,SAMPLE_SIZE
      DATA SCALE/0.01/
      REAL*4 DSI
      DATA DSI/2.321/

C     For idiot checks:
C     REAL*4 DUM,DIFPLD,STEMPDUM

      INTEGER*4 INDX
      DIMENSION INDX(TBINS)

C
```

```
C       NOTE: X,Y,Z assumed here to be in g/cm2!!!
C
C       Store relevant portion of input LET spectrum:
        M=0
        DO 4 I=1,N
           IF (L(I).LT.LMIN) GO TO 4
           IF (FLUX(I).LE.0.0) GOTO 6
           M=M+1
           LTEMP(M)=L(I)
           FTEMP(M)=FLUX(I)
 4      CONTINUE
 6      CONTINUE

C       Now we wish to do additional samplings around the peaks
C       in the pathlength distribution.
C       Specifically, the additional sampling is done on a scale
C       equal to 1% of the smallest dimension, but no larger than 0.01
C       microns.  The sampling is done at 100 points ranging from
C       x-10*scale to x+90*scale.

        PMAX=SQRT(X*X+Y*Y+Z*Z)
        SMALLEST=MIN(X,Y,Z)/0.0001/DSI
        SAMPLE_SIZE=SCALE*SMALLEST
        IF (SAMPLE_SIZE.GT.0.01) SAMPLE_SIZE=0.01

        DO 50 K=1,3
              IF (K.EQ.1) S=X
              IF (K.EQ.2) S=Y
              IF (K.EQ.3) S=Z
C             Suppress redundant samplings:
              IF (K.EQ.2 .and. (ABS(X-Y).LE.0.0001*X)) GOTO 50
              IF (K.EQ.3 .and.
     &        (ABS(X-Z).LE.0.0001*X .or. ABS(Y-Z).LE.0.0001*Y)) GOTO 50
              DO 45 I=-11,89
                 STEMP=S+FLOAT(I)*SAMPLE_SIZE*0.0001*DSI
                 IF (STEMP.LE.0. .or. STEMP.GT.PMAX) GOTO 45
C-----------------------------------------------------------------------
C                Idiot checks again:
C                 DUM=DIFPLD(STEMP,X,Y,Z)
C                 SDUM=S/DSI/0.0001
C                 STEMPDUM=STEMP/DSI/0.0001
C                 TYPE *,' S,I,STEMP(mics),DPLD: ',SDUM,I,STEMPDUM,DUM
C-----------------------------------------------------------------------
                 IF (M.LT.TBINS-1) THEN
                     M=M+1
                     LTEMP(M)=ENERGY/STEMP
                     XVAL=LTEMP(M)
                     CALL INTERPOLATE_INTLET(XVAL,N,L,FLUX,YVAL)
                     FTEMP(M)=YVAL
                 ENDIF
  45          CONTINUE
  50    CONTINUE

C
C       We now have the appropriate array of LET and FLUX values.
C       For the numerical integration, these must be ordered.
C       Use the INDEXX routine from Numerical Recipes:

        CALL INDEXX(M,TBINS,LTEMP,INDX)
```

```
      UPSET=ENERGY*AREA*3.1416*SUM

      RETURN
      END


C-----------------------------------------------------------------------

      SUBROUTINE TEMP_STORAGE(NBINS,NVAL,LVAL,FVAL,N,L,FLUX)

      IMPLICIT NONE
      INTEGER*4 NBINS,NVAL,N,K
      REAL*4 LVAL,FVAL,L,FLUX
      DIMENSION LVAL(1),FVAL(1),L(1),FLUX(1)

C     Copy Integral LET spectrum:
      DO 5 K=1,NVAL
              L(K)=LVAL(K)
              FLUX(K)=FVAL(K)
 5    CONTINUE
      N=NVAL

      IF (N.GT. NBINS) THEN
      WRITE(6,9999) N, NBINS
 9999 FORMAT('@ 10002 ABNORMAL TERMINATION: ',
     &   /,1x,' LET spec array out of bounds in GET_UPSET: ',
     &   /,1x,' N = ',I8,' NBINS = ',I8,
     &   /,1x,' STOP.')
      STOP
      ENDIF
      RETURN
      END

C-----------------------------------------------------------------------

      SUBROUTINE EXPAND_SAMPLING(NBINS,N,L,FLUX,
     &                           LMIN,ENERGY,PMAX,X,Y,Z,
     &                           M,LUSE,FUSE)

      IMPLICIT NONE
      INTEGER*4 NBINS,N,M,TBINS
      REAL*4 L,FLUX,LMIN,ENERGY,PMAX,X,Y,Z,LUSE,FUSE
      DIMENSION L(1),FLUX(1),LUSE(1),FUSE(1)
      PARAMETER (TBINS=5000)
      REAL*4 LTEMP,FTEMP
      DIMENSION LTEMP(TBINS),FTEMP(TBINS)
      INTEGER*4 I,K
      REAL*4 S,STEMP,SDUM,XVAL,YVAL
      REAL*4 SCALE,SMALLEST,SAMPLE_SIZE
      DATA SCALE/0.01/
      REAL*4 DSI
      DATA DSI/2.321/

C     For idiot checks:
C     REAL*4 DUM,DIFPLD,STEMPDUM

      INTEGER*4 INDX
      DIMENSION INDX(TBINS)

C
```

```
C       NOTE: X,Y,Z assumed here to be in g/cm2!!!
C
C       Store relevant portion of input LET spectrum:
        M=0
        DO 4 I=1,N
           IF (L(I).LT.LMIN) GO TO 4
           IF (FLUX(I).LE.0.0) GOTO 6
           M=M+1
           LTEMP(M)=L(I)
           FTEMP(M)=FLUX(I)
 4      CONTINUE
 6      CONTINUE

C       Now we wish to do additional samplings around the peaks
C       in the pathlength distribution.
C       Specifically, the additional sampling is done on a scale
C       equal to 1% of the smallest dimension, but no larger than 0.01
C       microns.  The sampling is done at 100 points ranging from
C       x-10*scale to x+90*scale.

        PMAX=SQRT(X*X+Y*Y+Z*Z)
        SMALLEST=MIN(X,Y,Z)/0.0001/DSI
        SAMPLE_SIZE=SCALE*SMALLEST
        IF (SAMPLE_SIZE.GT.0.01) SAMPLE_SIZE=0.01

        DO 50 K=1,3
              IF (K.EQ.1) S=X
              IF (K.EQ.2) S=Y
              IF (K.EQ.3) S=Z
C             Suppress redundant samplings:
              IF (K.EQ.2 .and. (ABS(X-Y).LE.0.0001*X)) GOTO 50
              IF (K.EQ.3 .and.
     &        (ABS(X-Z).LE.0.0001*X .or. ABS(Y-Z).LE.0.0001*Y)) GOTO 50
              DO 45 I=-11,89
                 STEMP=S+FLOAT(I)*SAMPLE_SIZE*0.0001*DSI
                 IF (STEMP.LE.0. .or. STEMP.GT.PMAX) GOTO 45
C-----------------------------------------------------------------------
C                Idiot checks again:
C                 DUM=DIFPLD(STEMP,X,Y,Z)
C                 SDUM=S/DSI/0.0001
C                 STEMPDUM=STEMP/DSI/0.0001
C                 TYPE *,' S,I,STEMP(mics),DPLD: ',SDUM,I,STEMPDUM,DUM
C-----------------------------------------------------------------------
                 IF (M.LT.TBINS-1) THEN
                    M=M+1
                    LTEMP(M)=ENERGY/STEMP
                    XVAL=LTEMP(M)
                    CALL INTERPOLATE_INTLET(XVAL,N,L,FLUX,YVAL)
                    FTEMP(M)=YVAL
                 ENDIF
 45           CONTINUE
 50     CONTINUE

C
C       We now have the appropriate array of LET and FLUX values.
C       For the numerical integration, these must be ordered.
C       Use the INDEXX routine from Numerical Recipes:

        CALL INDEXX(M,TBINS,LTEMP,INDX)
```

```
C        Now store the values according to increasing LET:
         DO 55 I=1,M
               IF (I.LE.NBINS.AND.I.LE.TBINS
     &                       .AND.INDX(I).LE.TBINS) THEN
                   LUSE(I)=LTEMP(INDX(I))
                   FUSE(I)=FTEMP(INDX(I))
               ELSE
                   WRITE(6,9999)  M,NBINS,TBINS,I,INDX(I)
 9999              FORMAT(' FATAL ERROR IN GET_UPSET:',
     &                 /,'       M,NBINS,TBINS: ',3I6,
     &                 /,'       I,INDX(I):     ',2I6)
               ENDIF
  55     CONTINUE
         RETURN
         END

         SUBROUTINE INTERPOLATE_INTLET(X,N,L,FLUX,Y)
C
C        Does a linear interpolation on a log-log plot of the
C        integral flux vs. LET curve.
C
         IMPLICIT NONE
         INTEGER*4 N
         REAL*4 X,Y,L,FLUX
         DIMENSION L(1),FLUX(1)
         REAL*8 X1,X2,X3,Y1,Y2,Y3,SLOPE
         INTEGER*4 I

         IF (X.LE.L(1)) THEN
             Y=FLUX(1)
         ELSEIF (X.GT.L(N)) THEN
C            Assume integral flux vanishes above the highest L value.
             Y=0.0
         ELSE
             DO 100 I=1,N-1
                IF (X.LE.L(I+1)) THEN
                    IF (FLUX(I).GT.0. .and. FLUX(I+1).GT.0.) THEN
                        X1=ALOG(L(I))
                        Y1=ALOG(FLUX(I))
                        X2=ALOG(L(I+1))
                        Y2=ALOG(FLUX(I+1))
                        SLOPE=(Y2-Y1)/(X2-X1)
                        X3=ALOG(X)
                        Y3=SLOPE*(X3-X1)+Y1
                        Y=EXP(Y3)
                        GOTO 150
                    ELSE
                        Y=0.0
                    ENDIF
                ENDIF
 100         CONTINUE
 150         CONTINUE
         ENDIF
         RETURN
         END
```

```
      LOGICAL FUNCTION INIGRID
C
C
C     This version uses the Epoch 1980.0 vertical cutoff grid of
C     Shea and Smart.
C
      IMPLICIT NONE

      INTEGER I,J
      REAL CUTOFF(33,72),CN,CS

      COMMON/CUTOFF80/CUTOFF,CN,CS
C
C     This common block contains the table of world wide vertical
C     geomagnetic cutoffs at 20 km altitude, tabulated every 5 degrees
C     in latitude (to +/- 80 degrees) and 5 degrees in longitude.  It
C     was given to JHA by private communication from D.F. Smart on 11/27/89.
C     This calculation is for Epoch 1980.0, ie. the cutoff calculation used
C     the 10th degree IGRF model (1980), as discussed in Shea & Smart, Proc.
C     18th ICRC, v. 3, p. 415 (1983).

       DATA (CUTOFF(1,J),J=1,72)/
     #    0.34,  0.31,  0.29,  0.26,  0.23,  0.21,  0.18,  0.16,  0.14,
     #    0.12,  0.10,  0.09,  0.07,  0.05,  0.04,  0.02,  0.01,  0.01,
     #    0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,
     #    0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.01,  0.01,
     #    0.02,  0.04,  0.05,  0.07,  0.09,  0.10,  0.12,  0.15,  0.17,
     #    0.20,  0.23,  0.25,  0.28,  0.32,  0.35,  0.39,  0.41,  0.43,
     #    0.45,  0.47,  0.50,  0.52,  0.53,  0.53,  0.54,  0.54,  0.53,
     #    0.53,  0.52,  0.51,  0.50,  0.46,  0.43,  0.39,  0.37,  0.36/
       DATA (CUTOFF(2,J),J=1,72)/
     #    0.59,  0.51,  0.45,  0.42,  0.37,  0.34,  0.28,  0.25,  0.21,
     #    0.18,  0.15,  0.12,  0.10,  0.08,  0.06,  0.04,  0.02,  0.00,
     #    0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,
     #    0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.02,
     #    0.04,  0.07,  0.09,  0.11,  0.15,  0.18,  0.22,  0.24,  0.30,
     #    0.34,  0.42,  0.48,  0.53,  0.60,  0.65,  0.70,  0.75,  0.83,
     #    0.87,  0.91,  0.90,  1.01,  1.00,  1.07,  1.04,  1.03,  1.03,
     #    0.98,  0.93,  0.93,  0.86,  0.82,  0.79,  0.72,  0.66,  0.60/
       DATA (CUTOFF(3,J),J=1,72)/
     #    0.84,  0.79,  0.73,  0.66,  0.58,  0.50,  0.46,  0.39,  0.36,
     #    0.30,  0.24,  0.21,  0.16,  0.13,  0.10,  0.08,  0.05,  0.03,
     #    0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,
     #    0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.02,  0.05,  0.08,
     #    0.10,  0.14,  0.18,  0.22,  0.27,  0.30,  0.40,  0.47,  0.53,
     #    0.60,  0.72,  0.80,  0.90,  1.06,  1.11,  1.20,  1.30,  1.43,
     #    1.51,  1.61,  1.64,  1.67,  1.72,  1.74,  1.75,  1.70,  1.65,
     #    1.59,  1.52,  1.42,  1.33,  1.27,  1.22,  1.11,  1.04,  0.95/
       DATA (CUTOFF(4,J),J=1,72)/
     #    1.29,  1.15,  1.03,  0.96,  0.85,  0.76,  0.72,  0.60,  0.57,
     #    0.48,  0.41,  0.35,  0.29,  0.25,  0.19,  0.14,  0.13,  0.09,
     #    0.06,  0.04,  0.02,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,
     #    0.00,  0.00,  0.02,  0.03,  0.05,  0.08,  0.10,  0.14,  0.18,
     #    0.22,  0.27,  0.35,  0.42,  0.49,  0.61,  0.69,  0.80,  0.94,
     #    1.05,  1.21,  1.34,  1.55,  1.65,  1.86,  1.96,  2.14,  2.32,
     #    2.46,  2.53,  2.67,  2.72,  2.70,  2.71,  2.63,  2.56,  2.51,
     #    2.40,  2.27,  2.20,  2.02,  1.87,  1.70,  1.61,  1.45,  1.35/
       DATA (CUTOFF(5,J),J=1,72)/
     #    1.69,  1.53,  1.45,  1.29,  1.15,  1.11,  1.03,  0.94,  0.83,
     #    0.76,  0.65,  0.59,  0.49,  0.42,  0.36,  0.30,  0.25,  0.20,
```

```
#     0.15,  0.13,  0.11,  0.09,  0.08,  0.07,  0.06,  0.06,  0.07,
#     0.08,  0.09,  0.12,  0.14,  0.16,  0.22,  0.26,  0.32,  0.39,
#     0.51,  0.58,  0.68,  0.76,  0.93,  1.07,  1.20,  1.38,  1.54,
#     1.77,  1.93,  2.10,  2.27,  2.64,  2.76,  2.98,  3.23,  3.49,
#     3.75,  3.79,  3.91,  4.01,  3.94,  3.94,  3.82,  3.71,  3.57,
#     3.39,  3.21,  2.98,  2.72,  2.52,  2.35,  2.14,  2.01,  1.78/
      DATA (CUTOFF(6,J),J=1,72)/
#     2.17,  1.97,  1.86,  1.72,  1.66,  1.56,  1.45,  1.30,  1.20,
#     1.14,  1.00,  0.92,  0.82,  0.73,  0.65,  0.56,  0.47,  0.41,
#     0.35,  0.31,  0.27,  0.24,  0.22,  0.21,  0.21,  0.21,  0.24,
#     0.25,  0.28,  0.32,  0.38,  0.43,  0.51,  0.59,  0.67,  0.80,
#     0.95,  1.07,  1.22,  1.36,  1.58,  1.76,  1.94,  2.18,  2.33,
#     2.64,  2.80,  3.03,  3.35,  3.80,  4.02,  4.29,  4.59,  4.75,
#     4.90,  5.07,  5.10,  5.18,  5.17,  5.00,  4.90,  4.74,  4.54,
#     4.25,  3.98,  3.70,  3.48,  3.21,  2.98,  2.77,  2.51,  2.32/
      DATA (CUTOFF(7,J),J=1,72)/
#     2.76,  2.55,  2.38,  2.27,  2.09,  2.02,  1.97,  1.85,  1.73,
#     1.64,  1.52,  1.42,  1.30,  1.19,  1.05,  0.94,  0.86,  0.78,
#     0.71,  0.67,  0.58,  0.56,  0.53,  0.53,  0.53,  0.51,  0.56,
#     0.60,  0.66,  0.71,  0.81,  0.89,  1.01,  1.15,  1.27,  1.42,
#     1.65,  1.83,  2.03,  2.24,  2.51,  2.70,  2.94,  3.15,  3.40,
#     3.79,  4.05,  4.20,  4.58,  4.85,  5.29,  5.61,  6.03,  6.62,
#     7.05,  7.36,  7.42,  7.41,  7.32,  7.10,  6.77,  6.37,  5.95,
#     5.47,  5.01,  4.60,  4.27,  4.00,  3.62,  3.42,  3.14,  2.87/
      DATA (CUTOFF(8,J),J=1,72)/
#     3.33,  3.22,  3.01,  2.88,  2.80,  2.67,  2.53,  2.50,  2.47,
#     2.38,  2.29,  2.10,  2.00,  1.82,  1.69,  1.54,  1.47,  1.35,
#     1.28,  1.18,  1.14,  1.12,  1.08,  1.08,  1.10,  1.13,  1.16,
#     1.22,  1.33,  1.36,  1.47,  1.67,  1.79,  2.01,  2.17,  2.43,
#     2.69,  2.96,  3.15,  3.30,  3.70,  4.08,  4.28,  4.42,  4.63,
#     4.99,  5.29,  5.61,  6.01,  6.63,  7.41,  7.87,  8.16,  8.69,
#     8.89,  8.97,  8.77,  8.52,  8.31,  8.01,  7.74,  7.58,  7.16,
#     6.80,  6.43,  5.89,  5.34,  4.85,  4.43,  4.15,  3.82,  3.52/
      DATA (CUTOFF(9,J),J=1,72)/
#     4.03,  3.85,  3.72,  3.62,  3.49,  3.41,  3.38,  3.33,  3.33,
#     3.26,  3.18,  3.00,  2.89,  2.77,  2.69,  2.58,  2.35,  2.25,
#     2.21,  2.06,  2.02,  2.04,  2.05,  2.05,  2.04,  2.01,  2.14,
#     2.22,  2.31,  2.43,  2.53,  2.86,  2.96,  3.29,  3.48,  3.74,
#     4.15,  4.48,  4.76,  4.84,  5.05,  5.30,  5.60,  5.96,  6.31,
#     6.76,  7.27,  7.91,  8.18,  8.55,  9.11,  9.73, 10.05, 10.14,
#    10.08,  9.98,  9.81,  9.63,  9.39,  9.13,  8.80,  8.32,  7.88,
#     7.35,  6.82,  6.68,  6.18,  5.94,  5.48,  5.00,  4.60,  4.34/
      DATA (CUTOFF(10,J),J=1,72)/
#     5.05,  4.79,  4.65,  4.49,  4.42,  4.39,  4.37,  4.33,  4.39,
#     4.38,  4.34,  4.26,  4.12,  4.12,  3.99,  3.85,  3.68,  3.52,
#     3.47,  3.40,  3.40,  3.41,  3.35,  3.35,  3.34,  3.37,  3.45,
#     3.55,  3.69,  3.89,  4.10,  4.35,  4.51,  4.90,  5.18,  5.38,
#     5.65,  5.95,  6.25,  6.62,  7.01,  7.47,  8.01,  8.51,  8.84,
#     8.73,  8.41,  9.00,  9.62, 10.73, 10.95, 11.05, 11.10, 11.12,
#    11.09, 10.95, 10.75, 10.55, 10.29, 10.01,  9.73,  9.39,  9.00,
#     8.50,  7.99,  7.47,  6.88,  6.52,  6.12,  5.95,  5.66,  5.32/
      DATA (CUTOFF(11,J),J=1,72)/
#     6.11,  5.99,  5.86,  5.84,  5.85,  5.83,  5.84,  5.82,  5.91,
#     5.80,  5.74,  5.72,  5.58,  5.51,  5.45,  5.40,  5.27,  5.23,
#     5.19,  5.14,  5.18,  5.14,  5.13,  5.06,  5.09,  5.21,  5.22,
#     5.38,  5.45,  5.60,  5.84,  6.02,  6.32,  6.60,  6.91,  7.46,
#     7.98,  8.58,  9.05,  9.45,  9.84,  9.25,  9.40,  9.59, 10.12,
#    10.75, 11.25, 11.69, 11.79, 11.86, 11.91, 11.92, 11.91, 11.87,
#    11.78, 11.67, 11.53, 11.34, 11.12, 10.85, 10.56, 10.29,  9.95,
#     9.57,  9.15,  8.65,  8.15,  7.62,  7.14,  6.88,  6.74,  6.33/
```

```
      DATA (CUTOFF(12,J),J=1,72)/
     #   7.28,  7.32,  7.35,  7.29,  7.45,  7.52,  7.63,  7.73,  7.82,
     #   7.93,  8.02,  8.03,  8.02,  8.02,  7.88,  7.71,  7.53,  7.37,
     #   7.28,  7.29,  7.30,  7.26,  7.35,  7.35,  7.42,  7.44,  7.55,
     #   7.64,  7.82,  8.06,  8.36,  8.75,  9.24,  9.49,  9.96, 10.40,
     #  10.19, 10.44, 10.76, 11.24, 11.61, 11.82, 11.98, 12.26, 12.48,
     #  12.56, 12.62, 12.65, 12.67, 12.68, 12.66, 12.63, 12.57, 12.49,
     #  12.38, 12.25, 12.10, 11.94, 11.76, 11.59, 11.40, 11.13, 10.84,
     #  10.52, 10.21,  9.79,  9.35,  8.95,  8.50,  8.04,  7.78,  7.51/
      DATA (CUTOFF(13,J),J=1,72)/
     #   8.87,  8.81,  8.79,  8.89,  8.99,  9.05,  9.26,  9.42,  9.60,
     #   9.74,  9.92, 10.02, 10.24, 10.42, 10.26, 10.45, 10.52, 10.53,
     #  10.55, 10.58, 10.64, 10.69, 10.73, 10.69, 10.75, 10.56, 10.63,
     #  10.66, 10.38, 10.49, 10.62, 10.90, 11.30, 11.87, 12.27, 12.56,
     #  12.72, 12.86, 13.02, 13.15, 13.23, 13.31, 13.36, 13.40, 13.41,
     #  13.42, 13.41, 13.38, 13.35, 13.30, 13.25, 13.17, 13.08, 12.98,
     #  12.85, 12.71, 12.57, 12.42, 12.28, 12.14, 11.99, 11.85, 11.69,
     #  11.47, 11.28, 10.84, 10.49, 10.14,  9.75,  9.47,  9.19,  9.05/
      DATA (CUTOFF(14,J),J=1,72)/
     #  10.45, 10.49, 10.61, 10.63, 10.75, 10.92, 11.08, 11.32, 11.57,
     #  11.75, 11.92, 12.13, 12.32, 12.48, 12.70, 12.80, 12.91, 12.99,
     #  13.06, 13.14, 13.20, 13.24, 13.23, 13.20, 13.17, 13.13, 13.13,
     #  13.18, 13.25, 13.32, 13.39, 13.48, 13.58, 13.69, 13.81, 13.93,
     #  14.01, 14.09, 14.14, 14.17, 14.17, 14.17, 14.15, 14.12, 14.08,
     #  14.03, 13.98, 13.91, 13.84, 13.76, 13.67, 13.57, 13.46, 13.33,
     #  13.19, 13.04, 12.90, 12.77, 12.65, 12.56, 12.47, 12.38, 12.28,
     #  12.15, 11.98, 11.78, 11.53, 11.32, 11.08, 10.79, 10.62, 10.50/
      DATA (CUTOFF(15,J),J=1,72)/
     #  11.86, 11.92, 12.01, 12.11, 12.23, 12.39, 12.57, 12.77, 12.99,
     #  13.23, 13.46, 13.69, 13.90, 14.09, 14.26, 14.40, 14.52, 14.62,
     #  14.71, 14.79, 14.85, 14.88, 14.89, 14.87, 14.84, 14.79, 14.73,
     #  14.69, 14.66, 14.65, 14.65, 14.68, 14.72, 14.77, 14.81, 14.86,
     #  14.87, 14.87, 14.86, 14.82, 14.78, 14.72, 14.66, 14.59, 14.51,
     #  14.43, 14.35, 14.25, 14.15, 14.05, 13.93, 13.81, 13.67, 13.52,
     #  13.37, 13.22, 13.08, 12.97, 12.89, 12.84, 12.81, 12.78, 12.75,
     #  12.69, 12.60, 12.49, 12.36, 12.21, 12.05, 11.92, 11.85, 11.85/
      DATA (CUTOFF(16,J),J=1,72)/
     #  12.94, 13.03, 13.14, 13.27, 13.42, 13.58, 13.77, 13.97, 14.18,
     #  14.41, 14.64, 14.88, 15.10, 15.30, 15.48, 15.64, 15.77, 15.88,
     #  15.97, 16.03, 16.07, 16.07, 16.05, 16.01, 15.94, 15.86, 15.77,
     #  15.68, 15.61, 15.54, 15.50, 15.46, 15.44, 15.42, 15.41, 15.39,
     #  15.35, 15.31, 15.25, 15.17, 15.09, 15.01, 14.92, 14.82, 14.72,
     #  14.62, 14.51, 14.40, 14.28, 14.15, 14.01, 13.86, 13.70, 13.53,
     #  13.36, 13.21, 13.08, 12.99, 12.95, 12.95, 12.98, 13.03, 13.07,
     #  13.09, 13.08, 13.05, 13.00, 12.93, 12.88, 12.85, 12.84, 12.87/
      DATA (CUTOFF(17,J),J=1,72)/
     #  13.80, 13.93, 14.07, 14.22, 14.38, 14.55, 14.73, 14.92, 15.12,
     #  15.34, 15.56, 15.78, 16.00, 16.21, 16.39, 16.56, 16.69, 16.79,
     #  16.87, 16.91, 16.92, 16.90, 16.85, 16.77, 16.67, 16.56, 16.43,
     #  16.31, 16.18, 16.07, 15.97, 15.89, 15.81, 15.74, 15.67, 15.60,
     #  15.53, 15.44, 15.35, 15.25, 15.15, 15.04, 14.93, 14.83, 14.71,
     #  14.60, 14.48, 14.35, 14.21, 14.06, 13.89, 13.71, 13.52, 13.32,
     #  13.14, 12.98, 12.86, 12.80, 12.80, 12.87, 12.97, 13.09, 13.21,
     #  13.32, 13.40, 13.45, 13.48, 13.50, 13.52, 13.55, 13.61, 13.69/
      DATA (CUTOFF(18,J),J=1,72)/
     #  14.41, 14.56, 14.72, 14.88, 15.05, 15.21, 15.38, 15.56, 15.74,
     #  15.94, 16.14, 16.36, 16.57, 16.77, 16.95, 17.11, 17.25, 17.34,
     #  17.41, 17.43, 17.42, 17.38, 17.30, 17.20, 17.07, 16.93, 16.77,
     #  16.61, 16.45, 16.30, 16.15, 16.01, 15.89, 15.77, 15.66, 15.55,
     #  15.43, 15.31, 15.19, 15.07, 14.96, 14.84, 14.73, 14.61, 14.49,
```

```
     #    14.36, 14.22, 14.08, 13.92, 13.72, 13.51, 13.22, 12.99, 12.71,
     #    12.45, 12.34, 12.34, 12.29, 12.41, 12.55, 12.73, 12.93, 13.14,
     #    13.33, 13.50, 13.64, 13.76, 13.85, 13.94, 14.04, 14.14, 14.27/
      DATA (CUTOFF(19,J),J=1,72)/
     #    14.70, 14.86, 15.02, 15.19, 15.35, 15.51, 15.67, 15.82, 15.99,
     #    16.17, 16.35, 16.55, 16.75, 16.95, 17.13, 17.29, 17.42, 17.52,
     #    17.57, 17.59, 17.56, 17.50, 17.41, 17.29, 17.15, 16.99, 16.81,
     #    16.63, 16.44, 16.25, 16.06, 15.89, 15.72, 15.56, 15.41, 15.25,
     #    15.10, 14.96, 14.81, 14.68, 14.55, 14.42, 14.30, 14.18, 14.03,
     #    13.88, 13.74, 13.56, 13.35, 13.13, 12.75, 12.36, 12.00, 11.59,
     #    11.27, 11.05, 10.96, 11.06, 11.41, 11.85, 12.22, 12.53, 12.83,
     #    13.11, 13.36, 13.58, 13.78, 13.95, 14.10, 14.25, 14.39, 14.54/
      DATA (CUTOFF(20,J),J=1,72)/
     #    14.63, 14.79, 14.95, 15.10, 15.25, 15.40, 15.54, 15.68, 15.83,
     #    15.99, 16.16, 16.34, 16.54, 16.73, 16.91, 17.07, 17.20, 17.29,
     #    17.35, 17.36, 17.34, 17.27, 17.18, 17.06, 16.91, 16.75, 16.56,
     #    16.37, 16.16, 15.95, 15.75, 15.54, 15.34, 15.14, 14.95, 14.77,
     #    14.59, 14.41, 14.25, 14.09, 13.95, 13.81, 13.67, 13.52, 13.35,
     #    13.19, 12.99, 12.77, 12.46, 12.06, 11.60, 11.05, 10.41,  9.86,
     #     9.27,  8.69,  8.41,  8.64,  9.15, 10.07, 11.01, 11.75, 12.22,
     #    12.60, 12.94, 13.24, 13.50, 13.73, 13.94, 14.12, 14.30, 14.46/
      DATA (CUTOFF(21,J),J=1,72)/
     #    14.17, 14.32, 14.47, 14.61, 14.74, 14.87, 14.99, 15.12, 15.25,
     #    15.39, 15.55, 15.72, 15.91, 16.09, 16.27, 16.43, 16.57, 16.67,
     #    16.73, 16.75, 16.73, 16.68, 16.60, 16.49, 16.36, 16.21, 16.03,
     #    15.85, 15.64, 15.43, 15.21, 14.99, 14.76, 14.54, 14.32, 14.11,
     #    13.90, 13.70, 13.51, 13.33, 13.16, 13.00, 12.84, 12.64, 12.45,
     #    12.23, 11.96, 11.67, 11.32, 10.89, 10.28,  9.35,  8.60,  8.31,
     #     7.67,  7.15,  6.79,  6.61,  7.24,  8.16,  8.49,  9.68, 10.89,
     #    11.67, 12.18, 12.55, 12.88, 13.17, 13.42, 13.63, 13.83, 14.00/
      DATA (CUTOFF(22,J),J=1,72)/
     #    13.25, 13.38, 13.53, 13.68, 13.81, 13.92, 14.03, 14.14, 14.25,
     #    14.38, 14.52, 14.68, 14.86, 15.04, 15.22, 15.37, 15.52, 15.62,
     #    15.69, 15.73, 15.73, 15.70, 15.65, 15.57, 15.47, 15.36, 15.22,
     #    15.05, 14.87, 14.67, 14.46, 14.23, 14.00, 13.76, 13.52, 13.28,
     #    13.04, 12.81, 12.58, 12.38, 12.17, 11.96, 11.74, 11.49, 11.21,
     #    10.88, 10.53, 10.10,  9.63,  9.06,  8.44,  7.74,  7.09,  6.49,
     #     6.02,  5.69,  5.59,  5.51,  5.78,  6.15,  7.02,  8.04,  9.04,
     #     9.98, 10.81, 11.34, 11.78, 12.14, 12.42, 12.68, 12.90, 13.07/
      DATA (CUTOFF(23,J),J=1,72)/
     #    11.49, 11.62, 11.74, 11.83, 11.91, 11.98, 12.10, 12.19, 12.35,
     #    12.51, 12.67, 12.83, 13.09, 13.34, 13.59, 13.82, 14.00, 14.11,
     #    14.19, 14.27, 14.30, 14.31, 14.31, 14.28, 14.23, 14.17, 14.08,
     #    13.97, 13.82, 13.66, 13.46, 13.25, 13.02, 12.75, 12.43, 12.07,
     #    11.69, 11.26, 10.79, 10.48, 10.14,  9.80,  9.65,  9.41,  9.10,
     #     8.78,  8.25,  7.62,  6.99,  6.40,  5.97,  5.54,  5.08,  4.74,
     #     4.38,  4.19,  4.10,  4.02,  4.21,  4.61,  5.14,  5.65,  6.46,
     #     7.54,  8.64,  9.38, 10.02, 10.52, 10.72, 10.94, 11.16, 11.33/
      DATA (CUTOFF(24,J),J=1,72)/
     #     9.77,  9.75,  9.86,  9.74,  9.76,  9.84, 10.01, 10.17, 10.27,
     #    10.42, 10.45, 10.72, 10.88, 11.20, 11.37, 11.27, 11.33, 11.34,
     #    11.39, 11.46, 11.56, 11.63, 11.74, 11.87, 11.95, 12.08, 12.17,
     #    12.18, 12.09, 11.92, 11.69, 11.37, 11.04, 10.67, 10.31,  9.93,
     #     9.52,  9.44,  9.49,  8.99,  8.52,  8.15,  7.67,  7.07,  6.56,
     #     6.11,  5.75,  5.48,  5.17,  4.89,  4.56,  4.21,  3.80,  3.44,
     #     3.24,  3.08,  2.98,  2.99,  3.13,  3.41,  3.83,  4.15,  4.74,
     #     5.26,  5.76,  6.45,  7.36,  8.14,  8.72,  9.11,  9.38,  9.59/
      DATA (CUTOFF(25,J),J=1,72)/
     #     6.95,  7.19,  7.34,  7.44,  7.50,  7.52,  7.59,  7.58,  7.63,
     #     7.73,  7.83,  7.91,  8.07,  8.24,  8.38,  8.54,  8.71,  8.87,
```

```
#      8.99,  9.03,  9.13,  9.23,  9.30,  9.46,  9.57,  9.79,  9.90,
#      9.99, 10.01,  9.99,  9.82,  9.63,  9.37,  9.05,  8.64,  8.31,
#      7.96,  7.41,  6.90,  6.47,  6.08,  5.78,  5.49,  5.27,  4.89,
#      4.60,  4.34,  4.02,  3.63,  3.30,  3.08,  2.77,  2.54,  2.30,
#      2.14,  2.09,  2.04,  2.05,  2.13,  2.27,  2.57,  2.86,  3.22,
#      3.79,  4.18,  4.63,  5.08,  5.41,  5.76,  6.15,  6.42,  6.72/
      DATA (CUTOFF(26,J),J=1,72)/
#      4.99,  5.02,  5.09,  5.20,  5.21,  5.34,  5.35,  5.36,  5.32,
#      5.44,  5.52,  5.53,  5.66,  5.71,  5.73,  5.81,  5.98,  6.03,
#      6.08,  6.16,  6.19,  6.31,  6.37,  6.45,  6.59,  6.75,  6.85,
#      6.96,  7.03,  6.99,  6.96,  6.88,  6.66,  6.38,  6.17,  5.83,
#      5.60,  5.35,  5.03,  4.85,  4.59,  4.40,  4.15,  3.77,  3.47,
#      3.16,  2.92,  2.66,  2.37,  2.10,  1.88,  1.75,  1.58,  1.45,
#      1.35,  1.28,  1.25,  1.27,  1.36,  1.47,  1.62,  1.87,  2.11,
#      2.47,  2.71,  3.06,  3.35,  3.80,  4.17,  4.35,  4.61,  4.68/
      DATA (CUTOFF(27,J),J=1,72)/
#      3.32,  3.35,  3.49,  3.59,  3.69,  3.76,  3.83,  3.86,  3.91,
#      3.94,  3.97,  4.03,  4.06,  4.11,  4.17,  4.20,  4.32,  4.32,
#      4.34,  4.36,  4.44,  4.45,  4.49,  4.64,  4.69,  4.85,  5.00,
#      5.00,  5.05,  4.98,  4.97,  4.86,  4.77,  4.69,  4.54,  4.47,
#      4.25,  4.08,  3.78,  3.46,  3.21,  3.00,  2.81,  2.51,  2.25,
#      2.05,  1.79,  1.64,  1.42,  1.28,  1.11,  0.98,  0.87,  0.81,
#      0.75,  0.71,  0.72,  0.73,  0.79,  0.87,  0.95,  1.10,  1.27,
#      1.43,  1.67,  1.92,  2.13,  2.33,  2.65,  2.85,  2.96,  3.18/
      DATA (CUTOFF(28,J),J=1,72)/
#      2.00,  2.08,  2.26,  2.29,  2.36,  2.41,  2.45,  2.47,  2.51,
#      2.53,  2.64,  2.64,  2.67,  2.68,  2.69,  2.73,  2.77,  2.78,
#      2.84,  2.80,  2.86,  2.93,  2.93,  3.03,  3.12,  3.19,  3.26,
#      3.31,  3.38,  3.38,  3.31,  3.30,  3.26,  3.15,  3.08,  3.02,
#      2.85,  2.69,  2.49,  2.28,  2.18,  1.96,  1.75,  1.55,  1.37,
#      1.23,  1.07,  0.90,  0.81,  0.70,  0.58,  0.49,  0.44,  0.41,
#      0.36,  0.36,  0.35,  0.36,  0.38,  0.46,  0.50,  0.58,  0.68,
#      0.80,  0.93,  1.05,  1.21,  1.35,  1.54,  1.70,  1.81,  1.96/
      DATA (CUTOFF(29,J),J=1,72)/
#      1.16,  1.23,  1.29,  1.36,  1.39,  1.44,  1.43,  1.52,  1.54,
#      1.59,  1.60,  1.57,  1.62,  1.64,  1.69,  1.68,  1.67,  1.69,
#      1.70,  1.74,  1.73,  1.80,  1.86,  1.89,  1.96,  1.99,  2.06,
#      2.05,  2.09,  2.09,  2.12,  2.14,  2.04,  2.06,  1.96,  1.86,
#      1.76,  1.63,  1.55,  1.38,  1.27,  1.16,  0.98,  0.89,  0.77,
#      0.65,  0.56,  0.46,  0.40,  0.32,  0.27,  0.23,  0.20,  0.17,
#      0.16,  0.15,  0.15,  0.16,  0.17,  0.21,  0.23,  0.27,  0.34,
#      0.39,  0.44,  0.53,  0.64,  0.72,  0.85,  0.90,  1.00,  1.09/
      DATA (CUTOFF(30,J),J=1,72)/
#      0.60,  0.62,  0.66,  0.69,  0.76,  0.77,  0.80,  0.82,  0.87,
#      0.87,  0.90,  0.88,  0.91,  0.93,  0.94,  0.94,  0.98,  0.98,
#      0.98,  1.00,  1.06,  1.03,  1.04,  1.09,  1.12,  1.14,  1.16,
#      1.23,  1.14,  1.22,  1.18,  1.18,  1.18,  1.11,  1.09,  1.02,
#      0.99,  0.90,  0.83,  0.77,  0.68,  0.57,  0.49,  0.45,  0.39,
#      0.31,  0.26,  0.21,  0.17,  0.13,  0.10,  0.09,  0.07,  0.06,
#      0.05,  0.05,  0.05,  0.05,  0.06,  0.07,  0.09,  0.11,  0.14,
#      0.17,  0.20,  0.25,  0.30,  0.33,  0.38,  0.43,  0.50,  0.54/
      DATA (CUTOFF(31,J),J=1,72)/
#      0.27,  0.29,  0.31,  0.34,  0.36,  0.38,  0.39,  0.42,  0.41,
#      0.44,  0.45,  0.47,  0.48,  0.49,  0.48,  0.51,  0.51,  0.50,
#      0.51,  0.54,  0.56,  0.55,  0.55,  0.56,  0.58,  0.57,  0.58,
#      0.60,  0.62,  0.61,  0.61,  0.59,  0.57,  0.57,  0.53,  0.51,
#      0.47,  0.45,  0.38,  0.34,  0.30,  0.27,  0.23,  0.19,  0.16,
#      0.13,  0.10,  0.08,  0.06,  0.04,  0.02,  0.00,  0.00,  0.00,
#      0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.02,  0.04,
#      0.06,  0.07,  0.09,  0.11,  0.14,  0.16,  0.19,  0.22,  0.25/
```

```
      DATA (CUTOFF(32,J),J=1,72)/
     #   0.10,  0.11,  0.13,  0.14,  0.15,  0.16,  0.17,  0.19,  0.19,
     #   0.20,  0.21,  0.22,  0.22,  0.23,  0.24,  0.22,  0.24,  0.24,
     #   0.25,  0.25,  0.25,  0.25,  0.26,  0.28,  0.26,  0.27,  0.27,
     #   0.26,  0.27,  0.27,  0.26,  0.25,  0.25,  0.24,  0.22,  0.20,
     #   0.19,  0.18,  0.15,  0.14,  0.12,  0.10,  0.09,  0.07,  0.05,
     #   0.03,  0.02,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,
     #   0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,
     #   0.00,  0.00,  0.00,  0.02,  0.04,  0.05,  0.07,  0.08,  0.09/
      DATA (CUTOFF(33,J),J=1,72)/
     #   0.02,  0.03,  0.04,  0.05,  0.05,  0.06,  0.06,  0.07,  0.07,
     #   0.08,  0.08,  0.09,  0.09,  0.09,  0.09,  0.09,  0.09,  0.10,
     #   0.10,  0.10,  0.10,  0.10,  0.10,  0.11,  0.11,  0.11,  0.10,
     #   0.10,  0.10,  0.10,  0.10,  0.09,  0.09,  0.08,  0.08,  0.07,
     #   0.07,  0.06,  0.04,  0.03,  0.02,  0.01,  0.00,  0.00,  0.00,
     #   0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,
     #   0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,
     #   0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.00,  0.01,  0.01/

      DATA CN,CS/  0.05,  0.21/

      INIGRID=.TRUE.

      RETURN
      END

      SUBROUTINE InitPreCalcs(RigBins)
C
C
      IMPLICIT NONE

      INTEGER I,J,Nrigs
      PARAMETER (Nrigs=1001)

      REAL RigBins(Nrigs),RIGPC(Nrigs)
      REAL PCGTF1(Nrigs),PCGTF2(Nrigs),PCGTF3(Nrigs),PCGTF4(Nrigs)

      COMMON/PreCalcCMN/PCGTF1,PCGTF2,PCGTF3,PCGTF4

      DATA (RIGPC(I),I=1,90)/
     #  0.000, 0.020, 0.040, 0.060, 0.080, 0.100, 0.120, 0.140, 0.160,
     #  0.180, 0.200, 0.220, 0.240, 0.260, 0.280, 0.300, 0.320, 0.340,
     #  0.360, 0.380, 0.400, 0.420, 0.440, 0.460, 0.480, 0.500, 0.520,
     #  0.540, 0.560, 0.580, 0.600, 0.620, 0.640, 0.660, 0.680, 0.700,
     #  0.720, 0.740, 0.760, 0.780, 0.800, 0.820, 0.840, 0.860, 0.880,
     #  0.900, 0.920, 0.940, 0.960, 0.980, 1.000, 1.020, 1.040, 1.060,
     #  1.080, 1.100, 1.120, 1.140, 1.160, 1.180, 1.200, 1.220, 1.240,
     #  1.260, 1.280, 1.300, 1.320, 1.340, 1.360, 1.380, 1.400, 1.420,
     #  1.440, 1.460, 1.480, 1.500, 1.520, 1.540, 1.560, 1.580, 1.600,
     #  1.620, 1.640, 1.660, 1.680, 1.700, 1.720, 1.740, 1.760, 1.780/
      DATA (RIGPC(I),I=91,180)/
     #  1.800, 1.820, 1.840, 1.860, 1.880, 1.900, 1.920, 1.940, 1.960,
     #  1.980, 2.000, 2.020, 2.040, 2.060, 2.080, 2.100, 2.120, 2.140,
     #  2.160, 2.180, 2.200, 2.220, 2.240, 2.260, 2.280, 2.300, 2.320,
     #  2.340, 2.360, 2.380, 2.400, 2.420, 2.440, 2.460, 2.480, 2.500,
     #  2.520, 2.540, 2.560, 2.580, 2.600, 2.620, 2.640, 2.660, 2.680,
     #  2.700, 2.720, 2.740, 2.760, 2.780, 2.800, 2.820, 2.840, 2.860,
     #  2.880, 2.900, 2.920, 2.940, 2.960, 2.980, 3.000, 3.020, 3.040,
     #  3.060, 3.080, 3.100, 3.120, 3.140, 3.160, 3.180, 3.200, 3.220,
     #  3.240, 3.260, 3.280, 3.300, 3.320, 3.340, 3.360, 3.380, 3.400,
```

```
#     3.420, 3.440, 3.460, 3.480, 3.500, 3.520, 3.540, 3.560, 3.580/
      DATA (RIGPC(I),I=181,270)/
#     3.600, 3.620, 3.640, 3.660, 3.680, 3.700, 3.720, 3.740, 3.760,
#     3.780, 3.800, 3.820, 3.840, 3.860, 3.880, 3.900, 3.920, 3.940,
#     3.960, 3.980, 4.000, 4.020, 4.040, 4.060, 4.080, 4.100, 4.120,
#     4.140, 4.160, 4.180, 4.200, 4.220, 4.240, 4.260, 4.280, 4.300,
#     4.320, 4.340, 4.360, 4.380, 4.400, 4.420, 4.440, 4.460, 4.480,
#     4.500, 4.520, 4.540, 4.560, 4.580, 4.600, 4.620, 4.640, 4.660,
#     4.680, 4.700, 4.720, 4.740, 4.760, 4.780, 4.800, 4.820, 4.840,
#     4.860, 4.880, 4.900, 4.920, 4.940, 4.960, 4.980, 5.000, 5.020,
#     5.040, 5.060, 5.080, 5.100, 5.120, 5.140, 5.160, 5.180, 5.200,
#     5.220, 5.240, 5.260, 5.280, 5.300, 5.320, 5.340, 5.360, 5.380/
      DATA (RIGPC(I),I=271,360)/
#     5.400, 5.420, 5.440, 5.460, 5.480, 5.500, 5.520, 5.540, 5.560,
#     5.580, 5.600, 5.620, 5.640, 5.660, 5.680, 5.700, 5.720, 5.740,
#     5.760, 5.780, 5.800, 5.820, 5.840, 5.860, 5.880, 5.900, 5.920,
#     5.940, 5.960, 5.980, 6.000, 6.020, 6.040, 6.060, 6.080, 6.100,
#     6.120, 6.140, 6.160, 6.180, 6.200, 6.220, 6.240, 6.260, 6.280,
#     6.300, 6.320, 6.340, 6.360, 6.380, 6.400, 6.420, 6.440, 6.460,
#     6.480, 6.500, 6.520, 6.540, 6.560, 6.580, 6.600, 6.620, 6.640,
#     6.660, 6.680, 6.700, 6.720, 6.740, 6.760, 6.780, 6.800, 6.820,
#     6.840, 6.860, 6.880, 6.900, 6.920, 6.940, 6.960, 6.980, 7.000,
#     7.020, 7.040, 7.060, 7.080, 7.100, 7.120, 7.140, 7.160, 7.180/
      DATA (RIGPC(I),I=361,450)/
#     7.200, 7.220, 7.240, 7.260, 7.280, 7.300, 7.320, 7.340, 7.360,
#     7.380, 7.400, 7.420, 7.440, 7.460, 7.480, 7.500, 7.520, 7.540,
#     7.560, 7.580, 7.600, 7.620, 7.640, 7.660, 7.680, 7.700, 7.720,
#     7.740, 7.760, 7.780, 7.800, 7.820, 7.840, 7.860, 7.880, 7.900,
#     7.920, 7.940, 7.960, 7.980, 8.000, 8.020, 8.040, 8.060, 8.080,
#     8.100, 8.120, 8.140, 8.160, 8.180, 8.200, 8.220, 8.240, 8.260,
#     8.280, 8.300, 8.320, 8.340, 8.360, 8.380, 8.400, 8.420, 8.440,
#     8.460, 8.480, 8.500, 8.520, 8.540, 8.560, 8.580, 8.600, 8.620,
#     8.640, 8.660, 8.680, 8.700, 8.720, 8.740, 8.760, 8.780, 8.800,
#     8.820, 8.840, 8.860, 8.880, 8.900, 8.920, 8.940, 8.960, 8.980/
      DATA (RIGPC(I),I=451,540)/
#     9.000, 9.020, 9.040, 9.060, 9.080, 9.100, 9.120, 9.140, 9.160,
#     9.180, 9.200, 9.220, 9.240, 9.260, 9.280, 9.300, 9.320, 9.340,
#     9.360, 9.380, 9.400, 9.420, 9.440, 9.460, 9.480, 9.500, 9.520,
#     9.540, 9.560, 9.580, 9.600, 9.620, 9.640, 9.660, 9.680, 9.700,
#     9.720, 9.740, 9.760, 9.780, 9.800, 9.820, 9.840, 9.860, 9.880,
#     9.900, 9.920, 9.940, 9.960, 9.980,10.000,10.020,10.040,10.060,
#    10.080,10.100,10.120,10.140,10.160,10.180,10.200,10.220,10.240,
#    10.260,10.280,10.300,10.320,10.340,10.360,10.380,10.400,10.420,
#    10.440,10.460,10.480,10.500,10.520,10.540,10.560,10.580,10.600,
#    10.620,10.640,10.660,10.680,10.700,10.720,10.740,10.760,10.780/
      DATA (RIGPC(I),I=541,630)/
#    10.800,10.820,10.840,10.860,10.880,10.900,10.920,10.940,10.960,
#    10.980,11.000,11.020,11.040,11.060,11.080,11.100,11.120,11.140,
#    11.160,11.180,11.200,11.220,11.240,11.260,11.280,11.300,11.320,
#    11.340,11.360,11.380,11.400,11.420,11.440,11.460,11.480,11.500,
#    11.520,11.540,11.560,11.580,11.600,11.620,11.640,11.660,11.680,
#    11.700,11.720,11.740,11.760,11.780,11.800,11.820,11.840,11.860,
#    11.880,11.900,11.920,11.940,11.960,11.980,12.000,12.020,12.040,
#    12.060,12.080,12.100,12.120,12.140,12.160,12.180,12.200,12.220,
#    12.240,12.260,12.280,12.300,12.320,12.340,12.360,12.380,12.400,
#    12.420,12.440,12.460,12.480,12.500,12.520,12.540,12.560,12.580/
      DATA (RIGPC(I),I=631,720)/
#    12.600,12.620,12.640,12.660,12.680,12.700,12.720,12.740,12.760,
#    12.780,12.800,12.820,12.840,12.860,12.880,12.900,12.920,12.940,
#    12.960,12.980,13.000,13.020,13.040,13.060,13.080,13.100,13.120,
```

```
     #    13.140,13.160,13.180,13.200,13.220,13.240,13.260,13.280,13.300,
     #    13.320,13.340,13.360,13.380,13.400,13.420,13.440,13.460,13.480,
     #    13.500,13.520,13.540,13.560,13.580,13.600,13.620,13.640,13.660,
     #    13.680,13.700,13.720,13.740,13.760,13.780,13.800,13.820,13.840,
     #    13.860,13.880,13.900,13.920,13.940,13.960,13.980,14.000,14.020,
     #    14.040,14.060,14.080,14.100,14.120,14.140,14.160,14.180,14.200,
     #    14.220,14.240,14.260,14.280,14.300,14.320,14.340,14.360,14.380/
          DATA (RIGPC(I),I=721,810)/
     #    14.400,14.420,14.440,14.460,14.480,14.500,14.520,14.540,14.560,
     #    14.580,14.600,14.620,14.640,14.660,14.680,14.700,14.720,14.740,
     #    14.760,14.780,14.800,14.820,14.840,14.860,14.880,14.900,14.920,
     #    14.940,14.960,14.980,15.000,15.020,15.040,15.060,15.080,15.100,
     #    15.120,15.140,15.160,15.180,15.200,15.220,15.240,15.260,15.280,
     #    15.300,15.320,15.340,15.360,15.380,15.400,15.420,15.440,15.460,
     #    15.480,15.500,15.520,15.540,15.560,15.580,15.600,15.620,15.640,
     #    15.660,15.680,15.700,15.720,15.740,15.760,15.780,15.800,15.820,
     #    15.840,15.860,15.880,15.900,15.920,15.940,15.960,15.980,16.000,
     #    16.020,16.040,16.060,16.080,16.100,16.120,16.140,16.160,16.180/
          DATA (RIGPC(I),I=811,900)/
     #    16.200,16.220,16.240,16.260,16.280,16.300,16.320,16.340,16.360,
     #    16.380,16.400,16.420,16.440,16.460,16.480,16.500,16.520,16.540,
     #    16.560,16.580,16.600,16.620,16.640,16.660,16.680,16.700,16.720,
     #    16.740,16.760,16.780,16.800,16.820,16.840,16.860,16.880,16.900,
     #    16.920,16.940,16.960,16.980,17.000,17.020,17.040,17.060,17.080,
     #    17.100,17.120,17.140,17.160,17.180,17.200,17.220,17.240,17.260,
     #    17.280,17.300,17.320,17.340,17.360,17.380,17.400,17.420,17.440,
     #    17.460,17.480,17.500,17.520,17.540,17.560,17.580,17.600,17.620,
     #    17.640,17.660,17.680,17.700,17.720,17.740,17.760,17.780,17.800,
     #    17.820,17.840,17.860,17.880,17.900,17.920,17.940,17.960,17.980/
          DATA (RIGPC(I),I=901,990)/
     #    18.000,18.020,18.040,18.060,18.080,18.100,18.120,18.140,18.160,
     #    18.180,18.200,18.220,18.240,18.260,18.280,18.300,18.320,18.340,
     #    18.360,18.380,18.400,18.420,18.440,18.460,18.480,18.500,18.520,
     #    18.540,18.560,18.580,18.600,18.620,18.640,18.660,18.680,18.700,
     #    18.720,18.740,18.760,18.780,18.800,18.820,18.840,18.860,18.880,
     #    18.900,18.920,18.940,18.960,18.980,19.000,19.020,19.040,19.060,
     #    19.080,19.100,19.120,19.140,19.160,19.180,19.200,19.220,19.240,
     #    19.260,19.280,19.300,19.320,19.340,19.360,19.380,19.400,19.420,
     #    19.440,19.460,19.480,19.500,19.520,19.540,19.560,19.580,19.600,
     #    19.620,19.640,19.660,19.680,19.700,19.720,19.740,19.760,19.780/
          DATA (RIGPC(I),I=991,Nrigs)/
     #    19.800,19.820,19.840,19.860,19.880,19.900,19.920,19.940,19.960,
     #    19.980,20.000/

          DATA (PCGTF1(I),I=1,50)/
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.2139E-03,0.5824E-03,0.1253E-02,0.2217E-02,
     #    0.3307E-02,0.4387E-02,0.5475E-02,0.6627E-02,0.7837E-02,
     #    0.9039E-02,0.1019E-01,0.1139E-01,0.1273E-01,0.1424E-01,
     #    0.1583E-01,0.1741E-01,0.1889E-01,0.2021E-01,0.2132E-01,
     #    0.2230E-01,0.2319E-01,0.2404E-01,0.2491E-01,0.2583E-01,
     #    0.2678E-01,0.2778E-01,0.2881E-01,0.2987E-01,0.3096E-01,
     #    0.3209E-01,0.3323E-01,0.3440E-01,0.3558E-01,0.3678E-01,
     #    0.3799E-01,0.3921E-01,0.4044E-01,0.4167E-01,0.4290E-01,
     #    0.4413E-01,0.4535E-01,0.4657E-01,0.4777E-01,0.4896E-01/
          DATA (PCGTF1(I),I=51,100)/
     #    0.5014E-01,0.5129E-01,0.5242E-01,0.5352E-01,0.5460E-01,
     #    0.5565E-01,0.5668E-01,0.5769E-01,0.5867E-01,0.5963E-01,
     #    0.6057E-01,0.6149E-01,0.6239E-01,0.6327E-01,0.6414E-01,
     #    0.6500E-01,0.6584E-01,0.6666E-01,0.6748E-01,0.6828E-01,
```

```
#    0.6908E-01,0.6986E-01,0.7064E-01,0.7141E-01,0.7218E-01,
#    0.7294E-01,0.7370E-01,0.7446E-01,0.7522E-01,0.7597E-01,
#    0.7673E-01,0.7749E-01,0.7825E-01,0.7902E-01,0.7979E-01,
#    0.8057E-01,0.8135E-01,0.8215E-01,0.8295E-01,0.8376E-01,
#    0.8459E-01,0.8543E-01,0.8628E-01,0.8715E-01,0.8803E-01,
#    0.8894E-01,0.8985E-01,0.9079E-01,0.9175E-01,0.9273E-01/
     DATA (PCGTF1(I),I=101,150)/
#    0.9373E-01,0.9475E-01,0.9579E-01,0.9684E-01,0.9791E-01,
#    0.9900E-01,0.1001E+00,0.1012E+00,0.1024E+00,0.1035E+00,
#    0.1047E+00,0.1059E+00,0.1071E+00,0.1083E+00,0.1095E+00,
#    0.1107E+00,0.1119E+00,0.1132E+00,0.1144E+00,0.1157E+00,
#    0.1170E+00,0.1183E+00,0.1195E+00,0.1208E+00,0.1221E+00,
#    0.1234E+00,0.1247E+00,0.1261E+00,0.1274E+00,0.1287E+00,
#    0.1300E+00,0.1313E+00,0.1326E+00,0.1340E+00,0.1353E+00,
#    0.1366E+00,0.1379E+00,0.1392E+00,0.1406E+00,0.1419E+00,
#    0.1432E+00,0.1445E+00,0.1458E+00,0.1471E+00,0.1483E+00,
#    0.1496E+00,0.1509E+00,0.1522E+00,0.1534E+00,0.1546E+00/
     DATA (PCGTF1(I),I=151,200)/
#    0.1559E+00,0.1571E+00,0.1583E+00,0.1595E+00,0.1607E+00,
#    0.1619E+00,0.1631E+00,0.1643E+00,0.1654E+00,0.1666E+00,
#    0.1677E+00,0.1689E+00,0.1700E+00,0.1711E+00,0.1723E+00,
#    0.1734E+00,0.1745E+00,0.1756E+00,0.1766E+00,0.1777E+00,
#    0.1788E+00,0.1799E+00,0.1809E+00,0.1820E+00,0.1830E+00,
#    0.1840E+00,0.1851E+00,0.1861E+00,0.1871E+00,0.1881E+00,
#    0.1891E+00,0.1901E+00,0.1911E+00,0.1921E+00,0.1930E+00,
#    0.1940E+00,0.1950E+00,0.1959E+00,0.1969E+00,0.1978E+00,
#    0.1987E+00,0.1997E+00,0.2006E+00,0.2015E+00,0.2024E+00,
#    0.2033E+00,0.2042E+00,0.2051E+00,0.2060E+00,0.2069E+00/
     DATA (PCGTF1(I),I=201,250)/
#    0.2078E+00,0.2087E+00,0.2095E+00,0.2104E+00,0.2113E+00,
#    0.2121E+00,0.2130E+00,0.2138E+00,0.2147E+00,0.2155E+00,
#    0.2163E+00,0.2172E+00,0.2180E+00,0.2188E+00,0.2196E+00,
#    0.2204E+00,0.2212E+00,0.2220E+00,0.2228E+00,0.2236E+00,
#    0.2244E+00,0.2252E+00,0.2260E+00,0.2268E+00,0.2276E+00,
#    0.2283E+00,0.2291E+00,0.2299E+00,0.2306E+00,0.2314E+00,
#    0.2322E+00,0.2329E+00,0.2337E+00,0.2344E+00,0.2352E+00,
#    0.2359E+00,0.2367E+00,0.2374E+00,0.2381E+00,0.2389E+00,
#    0.2396E+00,0.2403E+00,0.2411E+00,0.2418E+00,0.2425E+00,
#    0.2432E+00,0.2440E+00,0.2447E+00,0.2454E+00,0.2461E+00/
     DATA (PCGTF1(I),I=251,300)/
#    0.2468E+00,0.2475E+00,0.2483E+00,0.2490E+00,0.2497E+00,
#    0.2504E+00,0.2511E+00,0.2518E+00,0.2525E+00,0.2532E+00,
#    0.2539E+00,0.2546E+00,0.2553E+00,0.2560E+00,0.2567E+00,
#    0.2574E+00,0.2581E+00,0.2588E+00,0.2595E+00,0.2602E+00,
#    0.2609E+00,0.2616E+00,0.2622E+00,0.2629E+00,0.2636E+00,
#    0.2643E+00,0.2650E+00,0.2657E+00,0.2664E+00,0.2670E+00,
#    0.2677E+00,0.2684E+00,0.2691E+00,0.2698E+00,0.2704E+00,
#    0.2711E+00,0.2718E+00,0.2725E+00,0.2731E+00,0.2738E+00,
#    0.2745E+00,0.2752E+00,0.2758E+00,0.2765E+00,0.2772E+00,
#    0.2779E+00,0.2785E+00,0.2792E+00,0.2799E+00,0.2805E+00/
     DATA (PCGTF1(I),I=301,350)/
#    0.2812E+00,0.2819E+00,0.2826E+00,0.2832E+00,0.2839E+00,
#    0.2846E+00,0.2852E+00,0.2859E+00,0.2866E+00,0.2872E+00,
#    0.2879E+00,0.2886E+00,0.2892E+00,0.2899E+00,0.2906E+00,
#    0.2912E+00,0.2919E+00,0.2926E+00,0.2932E+00,0.2939E+00,
#    0.2946E+00,0.2952E+00,0.2959E+00,0.2966E+00,0.2972E+00,
#    0.2979E+00,0.2985E+00,0.2992E+00,0.2999E+00,0.3005E+00,
#    0.3012E+00,0.3019E+00,0.3025E+00,0.3032E+00,0.3039E+00,
#    0.3046E+00,0.3052E+00,0.3059E+00,0.3066E+00,0.3072E+00,
#    0.3079E+00,0.3086E+00,0.3092E+00,0.3099E+00,0.3106E+00,
```

```
#     0.3113E+00,0.3119E+00,0.3126E+00,0.3133E+00,0.3139E+00/
      DATA (PCGTF1(I),I=351,400)/
#     0.3146E+00,0.3153E+00,0.3160E+00,0.3167E+00,0.3173E+00,
#     0.3180E+00,0.3187E+00,0.3194E+00,0.3200E+00,0.3207E+00,
#     0.3214E+00,0.3221E+00,0.3228E+00,0.3235E+00,0.3241E+00,
#     0.3248E+00,0.3255E+00,0.3262E+00,0.3269E+00,0.3276E+00,
#     0.3283E+00,0.3290E+00,0.3297E+00,0.3304E+00,0.3311E+00,
#     0.3318E+00,0.3324E+00,0.3331E+00,0.3338E+00,0.3345E+00,
#     0.3352E+00,0.3360E+00,0.3367E+00,0.3374E+00,0.3381E+00,
#     0.3388E+00,0.3395E+00,0.3402E+00,0.3409E+00,0.3416E+00,
#     0.3423E+00,0.3430E+00,0.3437E+00,0.3445E+00,0.3452E+00,
#     0.3459E+00,0.3466E+00,0.3473E+00,0.3480E+00,0.3488E+00/
      DATA (PCGTF1(I),I=401,450)/
#     0.3495E+00,0.3502E+00,0.3509E+00,0.3516E+00,0.3524E+00,
#     0.3531E+00,0.3538E+00,0.3545E+00,0.3552E+00,0.3560E+00,
#     0.3567E+00,0.3574E+00,0.3581E+00,0.3589E+00,0.3596E+00,
#     0.3603E+00,0.3610E+00,0.3618E+00,0.3625E+00,0.3632E+00,
#     0.3639E+00,0.3647E+00,0.3654E+00,0.3661E+00,0.3668E+00,
#     0.3676E+00,0.3683E+00,0.3690E+00,0.3697E+00,0.3705E+00,
#     0.3712E+00,0.3719E+00,0.3726E+00,0.3734E+00,0.3741E+00,
#     0.3748E+00,0.3755E+00,0.3763E+00,0.3770E+00,0.3777E+00,
#     0.3784E+00,0.3792E+00,0.3799E+00,0.3806E+00,0.3813E+00,
#     0.3821E+00,0.3828E+00,0.3835E+00,0.3842E+00,0.3850E+00/
      DATA (PCGTF1(I),I=451,500)/
#     0.3857E+00,0.3864E+00,0.3871E+00,0.3878E+00,0.3886E+00,
#     0.3893E+00,0.3900E+00,0.3907E+00,0.3914E+00,0.3921E+00,
#     0.3929E+00,0.3936E+00,0.3943E+00,0.3950E+00,0.3957E+00,
#     0.3964E+00,0.3971E+00,0.3978E+00,0.3985E+00,0.3993E+00,
#     0.4000E+00,0.4007E+00,0.4014E+00,0.4021E+00,0.4028E+00,
#     0.4035E+00,0.4042E+00,0.4049E+00,0.4056E+00,0.4063E+00,
#     0.4070E+00,0.4077E+00,0.4084E+00,0.4091E+00,0.4097E+00,
#     0.4104E+00,0.4111E+00,0.4118E+00,0.4125E+00,0.4132E+00,
#     0.4139E+00,0.4146E+00,0.4152E+00,0.4159E+00,0.4166E+00,
#     0.4173E+00,0.4179E+00,0.4186E+00,0.4193E+00,0.4200E+00/
      DATA (PCGTF1(I),I=501,550)/
#     0.4206E+00,0.4209E+00,0.4212E+00,0.4215E+00,0.4218E+00,
#     0.4220E+00,0.4223E+00,0.4226E+00,0.4229E+00,0.4232E+00,
#     0.4234E+00,0.4237E+00,0.4240E+00,0.4243E+00,0.4246E+00,
#     0.4249E+00,0.4251E+00,0.4254E+00,0.4257E+00,0.4260E+00,
#     0.4263E+00,0.4266E+00,0.4268E+00,0.4271E+00,0.4274E+00,
#     0.4277E+00,0.4280E+00,0.4283E+00,0.4286E+00,0.4288E+00,
#     0.4291E+00,0.4294E+00,0.4297E+00,0.4300E+00,0.4303E+00,
#     0.4306E+00,0.4309E+00,0.4311E+00,0.4314E+00,0.4317E+00,
#     0.4320E+00,0.4323E+00,0.4326E+00,0.4329E+00,0.4332E+00,
#     0.4334E+00,0.4337E+00,0.4340E+00,0.4343E+00,0.4346E+00/
      DATA (PCGTF1(I),I=551,600)/
#     0.4349E+00,0.4352E+00,0.4355E+00,0.4358E+00,0.4361E+00,
#     0.4363E+00,0.4366E+00,0.4369E+00,0.4372E+00,0.4375E+00,
#     0.4378E+00,0.4381E+00,0.4384E+00,0.4387E+00,0.4390E+00,
#     0.4393E+00,0.4396E+00,0.4399E+00,0.4401E+00,0.4404E+00,
#     0.4407E+00,0.4410E+00,0.4413E+00,0.4416E+00,0.4419E+00,
#     0.4422E+00,0.4425E+00,0.4428E+00,0.4431E+00,0.4434E+00,
#     0.4437E+00,0.4440E+00,0.4443E+00,0.4446E+00,0.4449E+00,
#     0.4452E+00,0.4455E+00,0.4458E+00,0.4461E+00,0.4464E+00,
#     0.4467E+00,0.4469E+00,0.4472E+00,0.4475E+00,0.4478E+00,
#     0.4481E+00,0.4484E+00,0.4487E+00,0.4490E+00,0.4493E+00/
      DATA (PCGTF1(I),I=601,650)/
#     0.4496E+00,0.4499E+00,0.4502E+00,0.4505E+00,0.4508E+00,
#     0.4511E+00,0.4514E+00,0.4517E+00,0.4520E+00,0.4523E+00,
#     0.4526E+00,0.4530E+00,0.4533E+00,0.4536E+00,0.4539E+00,
```

```
#     0.4542E+00,0.4545E+00,0.4548E+00,0.4551E+00,0.4554E+00,
#     0.4557E+00,0.4560E+00,0.4563E+00,0.4566E+00,0.4569E+00,
#     0.4572E+00,0.4575E+00,0.4578E+00,0.4581E+00,0.4584E+00,
#     0.4587E+00,0.4590E+00,0.4593E+00,0.4596E+00,0.4600E+00,
#     0.4603E+00,0.4606E+00,0.4609E+00,0.4612E+00,0.4615E+00,
#     0.4618E+00,0.4621E+00,0.4624E+00,0.4627E+00,0.4630E+00,
#     0.4633E+00,0.4636E+00,0.4640E+00,0.4643E+00,0.4646E+00/
      DATA (PCGTF1(I),I=651,700)/
#     0.4649E+00,0.4652E+00,0.4655E+00,0.4658E+00,0.4661E+00,
#     0.4664E+00,0.4668E+00,0.4671E+00,0.4674E+00,0.4677E+00,
#     0.4680E+00,0.4683E+00,0.4686E+00,0.4689E+00,0.4692E+00,
#     0.4696E+00,0.4699E+00,0.4702E+00,0.4705E+00,0.4708E+00,
#     0.4711E+00,0.4714E+00,0.4718E+00,0.4721E+00,0.4724E+00,
#     0.4727E+00,0.4730E+00,0.4733E+00,0.4736E+00,0.4740E+00,
#     0.4743E+00,0.4746E+00,0.4749E+00,0.4752E+00,0.4755E+00,
#     0.4759E+00,0.4762E+00,0.4765E+00,0.4768E+00,0.4771E+00,
#     0.4775E+00,0.4778E+00,0.4781E+00,0.4784E+00,0.4787E+00,
#     0.4791E+00,0.4794E+00,0.4797E+00,0.4800E+00,0.4803E+00/
      DATA (PCGTF1(I),I=701,750)/
#     0.4807E+00,0.4810E+00,0.4813E+00,0.4816E+00,0.4819E+00,
#     0.4823E+00,0.4826E+00,0.4829E+00,0.4832E+00,0.4835E+00,
#     0.4839E+00,0.4842E+00,0.4845E+00,0.4848E+00,0.4852E+00,
#     0.4855E+00,0.4858E+00,0.4861E+00,0.4865E+00,0.4868E+00,
#     0.4871E+00,0.4874E+00,0.4878E+00,0.4881E+00,0.4884E+00,
#     0.4887E+00,0.4891E+00,0.4894E+00,0.4897E+00,0.4900E+00,
#     0.4904E+00,0.4907E+00,0.4910E+00,0.4913E+00,0.4917E+00,
#     0.4920E+00,0.4923E+00,0.4927E+00,0.4930E+00,0.4933E+00,
#     0.4936E+00,0.4940E+00,0.4943E+00,0.4946E+00,0.4950E+00,
#     0.4953E+00,0.4956E+00,0.4960E+00,0.4963E+00,0.4966E+00/
      DATA (PCGTF1(I),I=751,800)/
#     0.4969E+00,0.4973E+00,0.4976E+00,0.4979E+00,0.4983E+00,
#     0.4986E+00,0.4989E+00,0.4993E+00,0.4996E+00,0.4999E+00,
#     0.5003E+00,0.5006E+00,0.5009E+00,0.5013E+00,0.5016E+00,
#     0.5019E+00,0.5023E+00,0.5026E+00,0.5030E+00,0.5033E+00,
#     0.5036E+00,0.5040E+00,0.5043E+00,0.5046E+00,0.5050E+00,
#     0.5053E+00,0.5056E+00,0.5060E+00,0.5063E+00,0.5067E+00,
#     0.5070E+00,0.5073E+00,0.5077E+00,0.5080E+00,0.5083E+00,
#     0.5087E+00,0.5090E+00,0.5094E+00,0.5097E+00,0.5100E+00,
#     0.5104E+00,0.5107E+00,0.5111E+00,0.5114E+00,0.5117E+00,
#     0.5121E+00,0.5124E+00,0.5128E+00,0.5131E+00,0.5135E+00/
      DATA (PCGTF1(I),I=801,850)/
#     0.5138E+00,0.5141E+00,0.5145E+00,0.5148E+00,0.5152E+00,
#     0.5155E+00,0.5159E+00,0.5162E+00,0.5165E+00,0.5169E+00,
#     0.5172E+00,0.5176E+00,0.5179E+00,0.5183E+00,0.5186E+00,
#     0.5190E+00,0.5193E+00,0.5197E+00,0.5200E+00,0.5204E+00,
#     0.5207E+00,0.5210E+00,0.5214E+00,0.5217E+00,0.5221E+00,
#     0.5224E+00,0.5228E+00,0.5231E+00,0.5235E+00,0.5238E+00,
#     0.5242E+00,0.5245E+00,0.5249E+00,0.5252E+00,0.5256E+00,
#     0.5259E+00,0.5263E+00,0.5266E+00,0.5270E+00,0.5273E+00,
#     0.5277E+00,0.5280E+00,0.5284E+00,0.5287E+00,0.5291E+00,
#     0.5295E+00,0.5298E+00,0.5302E+00,0.5305E+00,0.5309E+00/
      DATA (PCGTF1(I),I=851,900)/
#     0.5312E+00,0.5316E+00,0.5319E+00,0.5323E+00,0.5326E+00,
#     0.5330E+00,0.5334E+00,0.5337E+00,0.5341E+00,0.5344E+00,
#     0.5348E+00,0.5351E+00,0.5355E+00,0.5358E+00,0.5362E+00,
#     0.5366E+00,0.5369E+00,0.5373E+00,0.5376E+00,0.5380E+00,
#     0.5384E+00,0.5387E+00,0.5391E+00,0.5394E+00,0.5398E+00,
#     0.5402E+00,0.5405E+00,0.5409E+00,0.5412E+00,0.5416E+00,
#     0.5420E+00,0.5423E+00,0.5427E+00,0.5430E+00,0.5434E+00,
#     0.5438E+00,0.5441E+00,0.5445E+00,0.5449E+00,0.5452E+00,
```

```
     #    0.5456E+00,0.5459E+00,0.5463E+00,0.5467E+00,0.5470E+00,
     #    0.5474E+00,0.5478E+00,0.5481E+00,0.5485E+00,0.5489E+00/
          DATA (PCGTF1(I),I=901,950)/
     #    0.5492E+00,0.5496E+00,0.5500E+00,0.5503E+00,0.5507E+00,
     #    0.5511E+00,0.5514E+00,0.5518E+00,0.5522E+00,0.5525E+00,
     #    0.5529E+00,0.5533E+00,0.5536E+00,0.5540E+00,0.5544E+00,
     #    0.5548E+00,0.5551E+00,0.5555E+00,0.5559E+00,0.5562E+00,
     #    0.5566E+00,0.5570E+00,0.5574E+00,0.5577E+00,0.5581E+00,
     #    0.5585E+00,0.5588E+00,0.5592E+00,0.5596E+00,0.5600E+00,
     #    0.5603E+00,0.5607E+00,0.5611E+00,0.5615E+00,0.5618E+00,
     #    0.5622E+00,0.5626E+00,0.5630E+00,0.5633E+00,0.5637E+00,
     #    0.5641E+00,0.5645E+00,0.5648E+00,0.5652E+00,0.5656E+00,
     #    0.5660E+00,0.5663E+00,0.5667E+00,0.5671E+00,0.5675E+00/
          DATA (PCGTF1(I),I=951,Nrigs)/
     #    0.5679E+00,0.5682E+00,0.5686E+00,0.5690E+00,0.5694E+00,
     #    0.5698E+00,0.5701E+00,0.5705E+00,0.5709E+00,0.5713E+00,
     #    0.5717E+00,0.5720E+00,0.5724E+00,0.5728E+00,0.5732E+00,
     #    0.5736E+00,0.5740E+00,0.5743E+00,0.5747E+00,0.5751E+00,
     #    0.5755E+00,0.5759E+00,0.5763E+00,0.5766E+00,0.5770E+00,
     #    0.5774E+00,0.5778E+00,0.5782E+00,0.5786E+00,0.5789E+00,
     #    0.5793E+00,0.5797E+00,0.5801E+00,0.5805E+00,0.5809E+00,
     #    0.5813E+00,0.5817E+00,0.5820E+00,0.5824E+00,0.5828E+00,
     #    0.5832E+00,0.5836E+00,0.5840E+00,0.5844E+00,0.5848E+00,
     #    0.5852E+00,0.5856E+00,0.5859E+00,0.5863E+00,0.5867E+00,
     #    0.5871E+00/
          DATA (PCGTF2(I),I=1,50)/
     #    0.4848E-01,0.4962E-01,0.5075E-01,0.5189E-01,0.5303E-01,
     #    0.5417E-01,0.5533E-01,0.5646E-01,0.5754E-01,0.5861E-01,
     #    0.5976E-01,0.6109E-01,0.6272E-01,0.6472E-01,0.6689E-01,
     #    0.6874E-01,0.6991E-01,0.7067E-01,0.7145E-01,0.7242E-01,
     #    0.7351E-01,0.7463E-01,0.7570E-01,0.7664E-01,0.7742E-01,
     #    0.7809E-01,0.7869E-01,0.7927E-01,0.7987E-01,0.8050E-01,
     #    0.8118E-01,0.8190E-01,0.8265E-01,0.8343E-01,0.8425E-01,
     #    0.8510E-01,0.8597E-01,0.8687E-01,0.8779E-01,0.8873E-01,
     #    0.8970E-01,0.9068E-01,0.9168E-01,0.9269E-01,0.9371E-01,
     #    0.9474E-01,0.9578E-01,0.9682E-01,0.9787E-01,0.9892E-01/
          DATA (PCGTF2(I),I=51,100)/
     #    0.9997E-01,0.1010E+00,0.1021E+00,0.1031E+00,0.1041E+00,
     #    0.1051E+00,0.1062E+00,0.1072E+00,0.1082E+00,0.1092E+00,
     #    0.1102E+00,0.1111E+00,0.1121E+00,0.1131E+00,0.1141E+00,
     #    0.1151E+00,0.1160E+00,0.1170E+00,0.1179E+00,0.1189E+00,
     #    0.1199E+00,0.1208E+00,0.1218E+00,0.1227E+00,0.1237E+00,
     #    0.1246E+00,0.1256E+00,0.1266E+00,0.1275E+00,0.1285E+00,
     #    0.1294E+00,0.1304E+00,0.1313E+00,0.1323E+00,0.1333E+00,
     #    0.1343E+00,0.1352E+00,0.1362E+00,0.1372E+00,0.1382E+00,
     #    0.1392E+00,0.1402E+00,0.1412E+00,0.1422E+00,0.1432E+00,
     #    0.1442E+00,0.1453E+00,0.1463E+00,0.1474E+00,0.1484E+00/
          DATA (PCGTF2(I),I=101,150)/
     #    0.1495E+00,0.1505E+00,0.1516E+00,0.1527E+00,0.1538E+00,
     #    0.1549E+00,0.1560E+00,0.1571E+00,0.1582E+00,0.1593E+00,
     #    0.1604E+00,0.1616E+00,0.1627E+00,0.1638E+00,0.1650E+00,
     #    0.1661E+00,0.1672E+00,0.1684E+00,0.1695E+00,0.1707E+00,
     #    0.1718E+00,0.1729E+00,0.1741E+00,0.1752E+00,0.1764E+00,
     #    0.1775E+00,0.1787E+00,0.1798E+00,0.1810E+00,0.1821E+00,
     #    0.1832E+00,0.1844E+00,0.1855E+00,0.1866E+00,0.1878E+00,
     #    0.1889E+00,0.1900E+00,0.1911E+00,0.1922E+00,0.1933E+00,
     #    0.1944E+00,0.1955E+00,0.1966E+00,0.1977E+00,0.1988E+00,
     #    0.1998E+00,0.2009E+00,0.2020E+00,0.2030E+00,0.2040E+00/
          DATA (PCGTF2(I),I=151,200)/
     #    0.2051E+00,0.2061E+00,0.2071E+00,0.2081E+00,0.2091E+00,
```

```
#     0.2101E+00,0.2111E+00,0.2121E+00,0.2130E+00,0.2140E+00,
#     0.2149E+00,0.2159E+00,0.2168E+00,0.2178E+00,0.2187E+00,
#     0.2196E+00,0.2205E+00,0.2214E+00,0.2223E+00,0.2232E+00,
#     0.2241E+00,0.2250E+00,0.2258E+00,0.2267E+00,0.2276E+00,
#     0.2284E+00,0.2293E+00,0.2301E+00,0.2310E+00,0.2318E+00,
#     0.2326E+00,0.2334E+00,0.2343E+00,0.2351E+00,0.2359E+00,
#     0.2367E+00,0.2375E+00,0.2382E+00,0.2390E+00,0.2398E+00,
#     0.2406E+00,0.2414E+00,0.2421E+00,0.2429E+00,0.2436E+00,
#     0.2444E+00,0.2451E+00,0.2459E+00,0.2466E+00,0.2473E+00/
      DATA (PCGTF2(I),I=201,250)/
#     0.2481E+00,0.2488E+00,0.2495E+00,0.2502E+00,0.2509E+00,
#     0.2517E+00,0.2524E+00,0.2531E+00,0.2538E+00,0.2545E+00,
#     0.2551E+00,0.2558E+00,0.2565E+00,0.2572E+00,0.2579E+00,
#     0.2586E+00,0.2592E+00,0.2599E+00,0.2606E+00,0.2612E+00,
#     0.2619E+00,0.2626E+00,0.2632E+00,0.2639E+00,0.2645E+00,
#     0.2652E+00,0.2658E+00,0.2665E+00,0.2671E+00,0.2678E+00,
#     0.2684E+00,0.2690E+00,0.2697E+00,0.2703E+00,0.2709E+00,
#     0.2716E+00,0.2722E+00,0.2728E+00,0.2735E+00,0.2741E+00,
#     0.2747E+00,0.2753E+00,0.2760E+00,0.2766E+00,0.2772E+00,
#     0.2778E+00,0.2785E+00,0.2791E+00,0.2797E+00,0.2803E+00/
      DATA (PCGTF2(I),I=251,300)/
#     0.2810E+00,0.2816E+00,0.2822E+00,0.2828E+00,0.2834E+00,
#     0.2840E+00,0.2847E+00,0.2853E+00,0.2859E+00,0.2865E+00,
#     0.2871E+00,0.2878E+00,0.2884E+00,0.2890E+00,0.2896E+00,
#     0.2902E+00,0.2908E+00,0.2915E+00,0.2921E+00,0.2927E+00,
#     0.2933E+00,0.2939E+00,0.2945E+00,0.2951E+00,0.2957E+00,
#     0.2964E+00,0.2970E+00,0.2976E+00,0.2982E+00,0.2988E+00,
#     0.2994E+00,0.3000E+00,0.3006E+00,0.3012E+00,0.3018E+00,
#     0.3024E+00,0.3030E+00,0.3036E+00,0.3042E+00,0.3048E+00,
#     0.3054E+00,0.3060E+00,0.3066E+00,0.3072E+00,0.3078E+00,
#     0.3084E+00,0.3090E+00,0.3096E+00,0.3102E+00,0.3108E+00/
      DATA (PCGTF2(I),I=301,350)/
#     0.3114E+00,0.3120E+00,0.3126E+00,0.3132E+00,0.3138E+00,
#     0.3144E+00,0.3150E+00,0.3155E+00,0.3161E+00,0.3167E+00,
#     0.3173E+00,0.3179E+00,0.3185E+00,0.3190E+00,0.3196E+00,
#     0.3202E+00,0.3208E+00,0.3213E+00,0.3219E+00,0.3225E+00,
#     0.3231E+00,0.3236E+00,0.3242E+00,0.3248E+00,0.3253E+00,
#     0.3259E+00,0.3265E+00,0.3270E+00,0.3276E+00,0.3282E+00,
#     0.3287E+00,0.3293E+00,0.3298E+00,0.3304E+00,0.3310E+00,
#     0.3315E+00,0.3321E+00,0.3326E+00,0.3332E+00,0.3337E+00,
#     0.3343E+00,0.3348E+00,0.3353E+00,0.3359E+00,0.3364E+00,
#     0.3370E+00,0.3375E+00,0.3380E+00,0.3386E+00,0.3391E+00/
      DATA (PCGTF2(I),I=351,400)/
#     0.3396E+00,0.3402E+00,0.3407E+00,0.3412E+00,0.3417E+00,
#     0.3423E+00,0.3428E+00,0.3433E+00,0.3438E+00,0.3444E+00,
#     0.3449E+00,0.3454E+00,0.3459E+00,0.3464E+00,0.3469E+00,
#     0.3474E+00,0.3479E+00,0.3484E+00,0.3489E+00,0.3494E+00,
#     0.3499E+00,0.3504E+00,0.3509E+00,0.3514E+00,0.3519E+00,
#     0.3524E+00,0.3529E+00,0.3534E+00,0.3538E+00,0.3543E+00,
#     0.3548E+00,0.3553E+00,0.3557E+00,0.3562E+00,0.3567E+00,
#     0.3572E+00,0.3576E+00,0.3581E+00,0.3586E+00,0.3590E+00,
#     0.3595E+00,0.3599E+00,0.3604E+00,0.3609E+00,0.3613E+00,
#     0.3618E+00,0.3622E+00,0.3627E+00,0.3631E+00,0.3636E+00/
      DATA (PCGTF2(I),I=401,450)/
#     0.3640E+00,0.3645E+00,0.3649E+00,0.3654E+00,0.3658E+00,
#     0.3662E+00,0.3667E+00,0.3671E+00,0.3676E+00,0.3680E+00,
#     0.3684E+00,0.3689E+00,0.3693E+00,0.3697E+00,0.3702E+00,
#     0.3706E+00,0.3710E+00,0.3714E+00,0.3719E+00,0.3723E+00,
#     0.3727E+00,0.3731E+00,0.3736E+00,0.3740E+00,0.3744E+00,
#     0.3748E+00,0.3752E+00,0.3757E+00,0.3761E+00,0.3765E+00,
```

```
     #    0.3769E+00,0.3773E+00,0.3778E+00,0.3782E+00,0.3786E+00,
     #    0.3790E+00,0.3794E+00,0.3798E+00,0.3802E+00,0.3807E+00,
     #    0.3811E+00,0.3815E+00,0.3819E+00,0.3823E+00,0.3827E+00,
     #    0.3831E+00,0.3835E+00,0.3839E+00,0.3844E+00,0.3848E+00/
          DATA (PCGTF2(I),I=451,500)/
     #    0.3852E+00,0.3856E+00,0.3860E+00,0.3864E+00,0.3868E+00,
     #    0.3872E+00,0.3876E+00,0.3880E+00,0.3884E+00,0.3889E+00,
     #    0.3893E+00,0.3897E+00,0.3901E+00,0.3905E+00,0.3909E+00,
     #    0.3913E+00,0.3917E+00,0.3921E+00,0.3925E+00,0.3930E+00,
     #    0.3934E+00,0.3938E+00,0.3942E+00,0.3946E+00,0.3950E+00,
     #    0.3954E+00,0.3958E+00,0.3963E+00,0.3967E+00,0.3971E+00,
     #    0.3975E+00,0.3979E+00,0.3983E+00,0.3987E+00,0.3992E+00,
     #    0.3996E+00,0.4000E+00,0.4004E+00,0.4008E+00,0.4013E+00,
     #    0.4017E+00,0.4021E+00,0.4025E+00,0.4030E+00,0.4034E+00,
     #    0.4038E+00,0.4042E+00,0.4047E+00,0.4051E+00,0.4055E+00/
          DATA (PCGTF2(I),I=501,550)/
     #    0.4060E+00,0.4063E+00,0.4066E+00,0.4069E+00,0.4072E+00,
     #    0.4075E+00,0.4078E+00,0.4081E+00,0.4084E+00,0.4087E+00,
     #    0.4090E+00,0.4093E+00,0.4096E+00,0.4100E+00,0.4103E+00,
     #    0.4106E+00,0.4109E+00,0.4112E+00,0.4115E+00,0.4118E+00,
     #    0.4121E+00,0.4124E+00,0.4127E+00,0.4131E+00,0.4134E+00,
     #    0.4137E+00,0.4140E+00,0.4143E+00,0.4146E+00,0.4149E+00,
     #    0.4152E+00,0.4156E+00,0.4159E+00,0.4162E+00,0.4165E+00,
     #    0.4168E+00,0.4171E+00,0.4174E+00,0.4178E+00,0.4181E+00,
     #    0.4184E+00,0.4187E+00,0.4190E+00,0.4193E+00,0.4197E+00,
     #    0.4200E+00,0.4203E+00,0.4206E+00,0.4209E+00,0.4212E+00/
          DATA (PCGTF2(I),I=551,600)/
     #    0.4216E+00,0.4219E+00,0.4222E+00,0.4225E+00,0.4228E+00,
     #    0.4231E+00,0.4235E+00,0.4238E+00,0.4241E+00,0.4244E+00,
     #    0.4247E+00,0.4251E+00,0.4254E+00,0.4257E+00,0.4260E+00,
     #    0.4264E+00,0.4267E+00,0.4270E+00,0.4273E+00,0.4276E+00,
     #    0.4280E+00,0.4283E+00,0.4286E+00,0.4289E+00,0.4293E+00,
     #    0.4296E+00,0.4299E+00,0.4302E+00,0.4306E+00,0.4309E+00,
     #    0.4312E+00,0.4315E+00,0.4319E+00,0.4322E+00,0.4325E+00,
     #    0.4328E+00,0.4332E+00,0.4335E+00,0.4338E+00,0.4341E+00,
     #    0.4345E+00,0.4348E+00,0.4351E+00,0.4355E+00,0.4358E+00,
     #    0.4361E+00,0.4364E+00,0.4368E+00,0.4371E+00,0.4374E+00/
          DATA (PCGTF2(I),I=601,650)/
     #    0.4378E+00,0.4381E+00,0.4384E+00,0.4388E+00,0.4391E+00,
     #    0.4394E+00,0.4397E+00,0.4401E+00,0.4404E+00,0.4407E+00,
     #    0.4411E+00,0.4414E+00,0.4417E+00,0.4421E+00,0.4424E+00,
     #    0.4427E+00,0.4431E+00,0.4434E+00,0.4437E+00,0.4441E+00,
     #    0.4444E+00,0.4447E+00,0.4451E+00,0.4454E+00,0.4458E+00,
     #    0.4461E+00,0.4464E+00,0.4468E+00,0.4471E+00,0.4474E+00,
     #    0.4478E+00,0.4481E+00,0.4485E+00,0.4488E+00,0.4491E+00,
     #    0.4495E+00,0.4498E+00,0.4502E+00,0.4505E+00,0.4508E+00,
     #    0.4512E+00,0.4515E+00,0.4519E+00,0.4522E+00,0.4525E+00,
     #    0.4529E+00,0.4532E+00,0.4536E+00,0.4539E+00,0.4542E+00/
          DATA (PCGTF2(I),I=651,700)/
     #    0.4546E+00,0.4549E+00,0.4553E+00,0.4556E+00,0.4560E+00,
     #    0.4563E+00,0.4566E+00,0.4570E+00,0.4573E+00,0.4577E+00,
     #    0.4580E+00,0.4584E+00,0.4587E+00,0.4591E+00,0.4594E+00,
     #    0.4598E+00,0.4601E+00,0.4605E+00,0.4608E+00,0.4611E+00,
     #    0.4615E+00,0.4618E+00,0.4622E+00,0.4625E+00,0.4629E+00,
     #    0.4632E+00,0.4636E+00,0.4639E+00,0.4643E+00,0.4646E+00,
     #    0.4650E+00,0.4653E+00,0.4657E+00,0.4660E+00,0.4664E+00,
     #    0.4667E+00,0.4671E+00,0.4675E+00,0.4678E+00,0.4682E+00,
     #    0.4685E+00,0.4689E+00,0.4692E+00,0.4696E+00,0.4699E+00,
     #    0.4703E+00,0.4706E+00,0.4710E+00,0.4713E+00,0.4717E+00/
          DATA (PCGTF2(I),I=701,750)/
```

```
#     0.4721E+00,0.4724E+00,0.4728E+00,0.4731E+00,0.4735E+00,
#     0.4738E+00,0.4742E+00,0.4746E+00,0.4749E+00,0.4753E+00,
#     0.4756E+00,0.4760E+00,0.4764E+00,0.4767E+00,0.4771E+00,
#     0.4774E+00,0.4778E+00,0.4782E+00,0.4785E+00,0.4789E+00,
#     0.4792E+00,0.4796E+00,0.4800E+00,0.4803E+00,0.4807E+00,
#     0.4810E+00,0.4814E+00,0.4818E+00,0.4821E+00,0.4825E+00,
#     0.4829E+00,0.4832E+00,0.4836E+00,0.4840E+00,0.4843E+00,
#     0.4847E+00,0.4851E+00,0.4854E+00,0.4858E+00,0.4862E+00,
#     0.4865E+00,0.4869E+00,0.4873E+00,0.4876E+00,0.4880E+00,
#     0.4884E+00,0.4887E+00,0.4891E+00,0.4895E+00,0.4898E+00/
      DATA (PCGTF2(I),I=751,800)/
#     0.4902E+00,0.4906E+00,0.4909E+00,0.4913E+00,0.4917E+00,
#     0.4921E+00,0.4924E+00,0.4928E+00,0.4932E+00,0.4935E+00,
#     0.4939E+00,0.4943E+00,0.4947E+00,0.4950E+00,0.4954E+00,
#     0.4958E+00,0.4962E+00,0.4965E+00,0.4969E+00,0.4973E+00,
#     0.4977E+00,0.4980E+00,0.4984E+00,0.4988E+00,0.4992E+00,
#     0.4995E+00,0.4999E+00,0.5003E+00,0.5007E+00,0.5010E+00,
#     0.5014E+00,0.5018E+00,0.5022E+00,0.5026E+00,0.5029E+00,
#     0.5033E+00,0.5037E+00,0.5041E+00,0.5045E+00,0.5048E+00,
#     0.5052E+00,0.5056E+00,0.5060E+00,0.5064E+00,0.5068E+00,
#     0.5071E+00,0.5075E+00,0.5079E+00,0.5083E+00,0.5087E+00/
      DATA (PCGTF2(I),I=801,850)/
#     0.5091E+00,0.5094E+00,0.5098E+00,0.5102E+00,0.5106E+00,
#     0.5110E+00,0.5114E+00,0.5117E+00,0.5121E+00,0.5125E+00,
#     0.5129E+00,0.5133E+00,0.5137E+00,0.5141E+00,0.5145E+00,
#     0.5148E+00,0.5152E+00,0.5156E+00,0.5160E+00,0.5164E+00,
#     0.5168E+00,0.5172E+00,0.5176E+00,0.5180E+00,0.5184E+00,
#     0.5187E+00,0.5191E+00,0.5195E+00,0.5199E+00,0.5203E+00,
#     0.5207E+00,0.5211E+00,0.5215E+00,0.5219E+00,0.5223E+00,
#     0.5227E+00,0.5231E+00,0.5235E+00,0.5239E+00,0.5243E+00,
#     0.5246E+00,0.5250E+00,0.5254E+00,0.5258E+00,0.5262E+00,
#     0.5266E+00,0.5270E+00,0.5274E+00,0.5278E+00,0.5282E+00/
      DATA (PCGTF2(I),I=851,900)/
#     0.5286E+00,0.5290E+00,0.5294E+00,0.5298E+00,0.5302E+00,
#     0.5306E+00,0.5310E+00,0.5314E+00,0.5318E+00,0.5322E+00,
#     0.5326E+00,0.5330E+00,0.5334E+00,0.5338E+00,0.5342E+00,
#     0.5346E+00,0.5350E+00,0.5354E+00,0.5358E+00,0.5363E+00,
#     0.5367E+00,0.5371E+00,0.5375E+00,0.5379E+00,0.5383E+00,
#     0.5387E+00,0.5391E+00,0.5395E+00,0.5399E+00,0.5403E+00,
#     0.5407E+00,0.5411E+00,0.5415E+00,0.5419E+00,0.5424E+00,
#     0.5428E+00,0.5432E+00,0.5436E+00,0.5440E+00,0.5444E+00,
#     0.5448E+00,0.5452E+00,0.5456E+00,0.5460E+00,0.5465E+00,
#     0.5469E+00,0.5473E+00,0.5477E+00,0.5481E+00,0.5485E+00/
      DATA (PCGTF2(I),I=901,950)/
#     0.5489E+00,0.5494E+00,0.5498E+00,0.5502E+00,0.5506E+00,
#     0.5510E+00,0.5514E+00,0.5518E+00,0.5523E+00,0.5527E+00,
#     0.5531E+00,0.5535E+00,0.5539E+00,0.5543E+00,0.5548E+00,
#     0.5552E+00,0.5556E+00,0.5560E+00,0.5564E+00,0.5569E+00,
#     0.5573E+00,0.5577E+00,0.5581E+00,0.5585E+00,0.5590E+00,
#     0.5594E+00,0.5598E+00,0.5602E+00,0.5607E+00,0.5611E+00,
#     0.5615E+00,0.5619E+00,0.5624E+00,0.5628E+00,0.5632E+00,
#     0.5636E+00,0.5641E+00,0.5645E+00,0.5649E+00,0.5653E+00,
#     0.5658E+00,0.5662E+00,0.5666E+00,0.5670E+00,0.5675E+00,
#     0.5679E+00,0.5683E+00,0.5688E+00,0.5692E+00,0.5696E+00/
      DATA (PCGTF2(I),I=951,Nrigs)/
#     0.5700E+00,0.5705E+00,0.5709E+00,0.5713E+00,0.5718E+00,
#     0.5722E+00,0.5726E+00,0.5731E+00,0.5735E+00,0.5739E+00,
#     0.5744E+00,0.5748E+00,0.5752E+00,0.5757E+00,0.5761E+00,
#     0.5765E+00,0.5770E+00,0.5774E+00,0.5778E+00,0.5783E+00,
#     0.5787E+00,0.5791E+00,0.5796E+00,0.5800E+00,0.5805E+00,
```

```
     # 0.5809E+00,0.5813E+00,0.5818E+00,0.5822E+00,0.5826E+00,
     # 0.5831E+00,0.5835E+00,0.5840E+00,0.5844E+00,0.5849E+00,
     # 0.5853E+00,0.5857E+00,0.5862E+00,0.5866E+00,0.5871E+00,
     # 0.5875E+00,0.5879E+00,0.5884E+00,0.5888E+00,0.5893E+00,
     # 0.5897E+00,0.5902E+00,0.5906E+00,0.5911E+00,0.5915E+00,
     # 0.5920E+00/
       DATA (PCGTF3(I),I=1,50)/
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00/
       DATA (PCGTF3(I),I=51,100)/
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00/
       DATA (PCGTF3(I),I=101,150)/
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     # 0.0000E+00,0.2036E-05,0.4190E-05,0.6583E-05,0.9331E-05,
     # 0.1256E-04,0.1637E-04,0.2091E-04,0.2627E-04,0.3258E-04/
       DATA (PCGTF3(I),I=151,200)/
     # 0.3997E-04,0.4852E-04,0.5826E-04,0.6919E-04,0.8129E-04,
     # 0.9459E-04,0.1091E-03,0.1247E-03,0.1415E-03,0.1595E-03,
     # 0.1787E-03,0.1991E-03,0.2206E-03,0.2433E-03,0.2673E-03,
     # 0.2928E-03,0.3200E-03,0.3492E-03,0.3805E-03,0.4142E-03,
     # 0.4506E-03,0.4898E-03,0.5321E-03,0.5776E-03,0.6267E-03,
     # 0.6795E-03,0.7362E-03,0.7973E-03,0.8631E-03,0.9339E-03,
     # 0.1010E-02,0.1092E-02,0.1180E-02,0.1275E-02,0.1377E-02,
     # 0.1486E-02,0.1602E-02,0.1726E-02,0.1859E-02,0.2001E-02,
     # 0.2151E-02,0.2311E-02,0.2481E-02,0.2661E-02,0.2852E-02,
     # 0.3053E-02,0.3266E-02,0.3490E-02,0.3726E-02,0.3975E-02/
       DATA (PCGTF3(I),I=201,250)/
     # 0.4237E-02,0.4511E-02,0.4798E-02,0.5097E-02,0.5406E-02,
     # 0.5727E-02,0.6057E-02,0.6397E-02,0.6746E-02,0.7102E-02,
     # 0.7466E-02,0.7837E-02,0.8214E-02,0.8597E-02,0.8984E-02,
     # 0.9376E-02,0.9772E-02,0.1017E-01,0.1057E-01,0.1097E-01,
     # 0.1138E-01,0.1178E-01,0.1219E-01,0.1259E-01,0.1299E-01,
     # 0.1339E-01,0.1379E-01,0.1419E-01,0.1458E-01,0.1498E-01,
     # 0.1537E-01,0.1577E-01,0.1617E-01,0.1657E-01,0.1697E-01,
     # 0.1738E-01,0.1779E-01,0.1821E-01,0.1864E-01,0.1907E-01,
     # 0.1951E-01,0.1996E-01,0.2041E-01,0.2088E-01,0.2136E-01,
```

```
#    0.2185E-01,0.2235E-01,0.2287E-01,0.2340E-01,0.2394E-01/
     DATA (PCGTF3(I),I=251,300)/
#    0.2450E-01,0.2508E-01,0.2567E-01,0.2628E-01,0.2690E-01,
#    0.2753E-01,0.2818E-01,0.2884E-01,0.2951E-01,0.3020E-01,
#    0.3089E-01,0.3160E-01,0.3231E-01,0.3304E-01,0.3377E-01,
#    0.3452E-01,0.3527E-01,0.3602E-01,0.3679E-01,0.3756E-01,
#    0.3834E-01,0.3912E-01,0.3990E-01,0.4069E-01,0.4149E-01,
#    0.4228E-01,0.4308E-01,0.4388E-01,0.4468E-01,0.4548E-01,
#    0.4629E-01,0.4709E-01,0.4789E-01,0.4869E-01,0.4949E-01,
#    0.5028E-01,0.5107E-01,0.5186E-01,0.5264E-01,0.5342E-01,
#    0.5419E-01,0.5496E-01,0.5572E-01,0.5648E-01,0.5722E-01,
#    0.5796E-01,0.5869E-01,0.5941E-01,0.6012E-01,0.6082E-01/
     DATA (PCGTF3(I),I=301,350)/
#    0.6151E-01,0.6219E-01,0.6285E-01,0.6351E-01,0.6416E-01,
#    0.6479E-01,0.6542E-01,0.6604E-01,0.6665E-01,0.6726E-01,
#    0.6785E-01,0.6844E-01,0.6902E-01,0.6960E-01,0.7017E-01,
#    0.7074E-01,0.7130E-01,0.7186E-01,0.7241E-01,0.7296E-01,
#    0.7351E-01,0.7406E-01,0.7460E-01,0.7515E-01,0.7569E-01,
#    0.7623E-01,0.7677E-01,0.7732E-01,0.7786E-01,0.7841E-01,
#    0.7896E-01,0.7951E-01,0.8006E-01,0.8062E-01,0.8118E-01,
#    0.8174E-01,0.8231E-01,0.8289E-01,0.8347E-01,0.8405E-01,
#    0.8465E-01,0.8525E-01,0.8586E-01,0.8647E-01,0.8710E-01,
#    0.8773E-01,0.8837E-01,0.8902E-01,0.8969E-01,0.9036E-01/
     DATA (PCGTF3(I),I=351,400)/
#    0.9105E-01,0.9174E-01,0.9245E-01,0.9317E-01,0.9390E-01,
#    0.9464E-01,0.9540E-01,0.9616E-01,0.9693E-01,0.9772E-01,
#    0.9851E-01,0.9931E-01,0.1001E+00,0.1009E+00,0.1018E+00,
#    0.1026E+00,0.1035E+00,0.1043E+00,0.1052E+00,0.1061E+00,
#    0.1070E+00,0.1078E+00,0.1087E+00,0.1097E+00,0.1106E+00,
#    0.1115E+00,0.1124E+00,0.1133E+00,0.1143E+00,0.1152E+00,
#    0.1162E+00,0.1171E+00,0.1181E+00,0.1191E+00,0.1201E+00,
#    0.1210E+00,0.1220E+00,0.1230E+00,0.1240E+00,0.1250E+00,
#    0.1260E+00,0.1270E+00,0.1280E+00,0.1290E+00,0.1301E+00,
#    0.1311E+00,0.1321E+00,0.1331E+00,0.1342E+00,0.1352E+00/
     DATA (PCGTF3(I),I=401,450)/
#    0.1362E+00,0.1373E+00,0.1383E+00,0.1394E+00,0.1404E+00,
#    0.1414E+00,0.1425E+00,0.1435E+00,0.1446E+00,0.1456E+00,
#    0.1467E+00,0.1478E+00,0.1488E+00,0.1499E+00,0.1509E+00,
#    0.1520E+00,0.1530E+00,0.1541E+00,0.1552E+00,0.1562E+00,
#    0.1573E+00,0.1584E+00,0.1594E+00,0.1605E+00,0.1616E+00,
#    0.1626E+00,0.1637E+00,0.1648E+00,0.1658E+00,0.1669E+00,
#    0.1680E+00,0.1690E+00,0.1701E+00,0.1712E+00,0.1722E+00,
#    0.1733E+00,0.1744E+00,0.1754E+00,0.1765E+00,0.1776E+00,
#    0.1786E+00,0.1797E+00,0.1807E+00,0.1818E+00,0.1829E+00,
#    0.1839E+00,0.1850E+00,0.1860E+00,0.1871E+00,0.1882E+00/
     DATA (PCGTF3(I),I=451,500)/
#    0.1892E+00,0.1903E+00,0.1913E+00,0.1924E+00,0.1934E+00,
#    0.1945E+00,0.1955E+00,0.1965E+00,0.1976E+00,0.1986E+00,
#    0.1997E+00,0.2007E+00,0.2017E+00,0.2027E+00,0.2038E+00,
#    0.2048E+00,0.2058E+00,0.2068E+00,0.2079E+00,0.2089E+00,
#    0.2099E+00,0.2109E+00,0.2119E+00,0.2129E+00,0.2139E+00,
#    0.2149E+00,0.2159E+00,0.2169E+00,0.2179E+00,0.2189E+00,
#    0.2198E+00,0.2208E+00,0.2218E+00,0.2227E+00,0.2237E+00,
#    0.2247E+00,0.2256E+00,0.2266E+00,0.2275E+00,0.2285E+00,
#    0.2294E+00,0.2304E+00,0.2313E+00,0.2322E+00,0.2331E+00,
#    0.2341E+00,0.2350E+00,0.2359E+00,0.2368E+00,0.2377E+00/
     DATA (PCGTF3(I),I=501,550)/
#    0.2386E+00,0.2395E+00,0.2404E+00,0.2412E+00,0.2421E+00,
#    0.2430E+00,0.2439E+00,0.2447E+00,0.2456E+00,0.2464E+00,
#    0.2473E+00,0.2481E+00,0.2490E+00,0.2498E+00,0.2506E+00,
```

```
     #     0.2515E+00,0.2523E+00,0.2531E+00,0.2539E+00,0.2547E+00,
     #     0.2555E+00,0.2563E+00,0.2571E+00,0.2579E+00,0.2587E+00,
     #     0.2595E+00,0.2603E+00,0.2611E+00,0.2619E+00,0.2626E+00,
     #     0.2634E+00,0.2642E+00,0.2649E+00,0.2657E+00,0.2664E+00,
     #     0.2672E+00,0.2680E+00,0.2687E+00,0.2694E+00,0.2702E+00,
     #     0.2709E+00,0.2717E+00,0.2724E+00,0.2731E+00,0.2739E+00,
     #     0.2746E+00,0.2753E+00,0.2760E+00,0.2768E+00,0.2775E+00/
           DATA (PCGTF3(I),I=551,600)/
     #     0.2782E+00,0.2789E+00,0.2796E+00,0.2803E+00,0.2810E+00,
     #     0.2817E+00,0.2824E+00,0.2831E+00,0.2838E+00,0.2845E+00,
     #     0.2852E+00,0.2859E+00,0.2866E+00,0.2873E+00,0.2880E+00,
     #     0.2887E+00,0.2894E+00,0.2900E+00,0.2907E+00,0.2914E+00,
     #     0.2921E+00,0.2928E+00,0.2934E+00,0.2941E+00,0.2948E+00,
     #     0.2955E+00,0.2961E+00,0.2968E+00,0.2975E+00,0.2982E+00,
     #     0.2988E+00,0.2995E+00,0.3002E+00,0.3008E+00,0.3015E+00,
     #     0.3022E+00,0.3028E+00,0.3035E+00,0.3042E+00,0.3048E+00,
     #     0.3055E+00,0.3062E+00,0.3069E+00,0.3075E+00,0.3082E+00,
     #     0.3089E+00,0.3095E+00,0.3102E+00,0.3109E+00,0.3115E+00/
           DATA (PCGTF3(I),I=601,650)/
     #     0.3122E+00,0.3129E+00,0.3135E+00,0.3142E+00,0.3149E+00,
     #     0.3156E+00,0.3162E+00,0.3169E+00,0.3176E+00,0.3182E+00,
     #     0.3189E+00,0.3196E+00,0.3203E+00,0.3209E+00,0.3216E+00,
     #     0.3223E+00,0.3230E+00,0.3237E+00,0.3243E+00,0.3250E+00,
     #     0.3257E+00,0.3264E+00,0.3270E+00,0.3277E+00,0.3284E+00,
     #     0.3291E+00,0.3298E+00,0.3304E+00,0.3311E+00,0.3318E+00,
     #     0.3325E+00,0.3332E+00,0.3339E+00,0.3345E+00,0.3352E+00,
     #     0.3359E+00,0.3366E+00,0.3373E+00,0.3379E+00,0.3386E+00,
     #     0.3393E+00,0.3400E+00,0.3407E+00,0.3414E+00,0.3421E+00,
     #     0.3427E+00,0.3434E+00,0.3441E+00,0.3448E+00,0.3455E+00/
           DATA (PCGTF3(I),I=651,700)/
     #     0.3462E+00,0.3469E+00,0.3475E+00,0.3482E+00,0.3489E+00,
     #     0.3496E+00,0.3503E+00,0.3510E+00,0.3517E+00,0.3523E+00,
     #     0.3530E+00,0.3537E+00,0.3544E+00,0.3551E+00,0.3558E+00,
     #     0.3565E+00,0.3572E+00,0.3578E+00,0.3585E+00,0.3592E+00,
     #     0.3599E+00,0.3606E+00,0.3613E+00,0.3620E+00,0.3627E+00,
     #     0.3634E+00,0.3640E+00,0.3647E+00,0.3654E+00,0.3661E+00,
     #     0.3668E+00,0.3675E+00,0.3682E+00,0.3689E+00,0.3695E+00,
     #     0.3702E+00,0.3709E+00,0.3716E+00,0.3723E+00,0.3730E+00,
     #     0.3737E+00,0.3743E+00,0.3750E+00,0.3757E+00,0.3764E+00,
     #     0.3771E+00,0.3778E+00,0.3785E+00,0.3792E+00,0.3798E+00/
           DATA (PCGTF3(I),I=701,750)/
     #     0.3805E+00,0.3812E+00,0.3819E+00,0.3826E+00,0.3833E+00,
     #     0.3839E+00,0.3846E+00,0.3853E+00,0.3860E+00,0.3867E+00,
     #     0.3874E+00,0.3880E+00,0.3887E+00,0.3894E+00,0.3901E+00,
     #     0.3908E+00,0.3915E+00,0.3921E+00,0.3928E+00,0.3935E+00,
     #     0.3942E+00,0.3949E+00,0.3955E+00,0.3962E+00,0.3969E+00,
     #     0.3976E+00,0.3983E+00,0.3989E+00,0.3996E+00,0.4003E+00,
     #     0.4010E+00,0.4016E+00,0.4023E+00,0.4030E+00,0.4037E+00,
     #     0.4044E+00,0.4050E+00,0.4057E+00,0.4064E+00,0.4070E+00,
     #     0.4077E+00,0.4084E+00,0.4091E+00,0.4097E+00,0.4104E+00,
     #     0.4111E+00,0.4117E+00,0.4124E+00,0.4131E+00,0.4138E+00/
           DATA (PCGTF3(I),I=751,800)/
     #     0.4144E+00,0.4151E+00,0.4158E+00,0.4164E+00,0.4171E+00,
     #     0.4178E+00,0.4184E+00,0.4191E+00,0.4197E+00,0.4204E+00,
     #     0.4211E+00,0.4217E+00,0.4224E+00,0.4231E+00,0.4237E+00,
     #     0.4244E+00,0.4250E+00,0.4257E+00,0.4264E+00,0.4270E+00,
     #     0.4277E+00,0.4283E+00,0.4290E+00,0.4296E+00,0.4303E+00,
     #     0.4309E+00,0.4316E+00,0.4322E+00,0.4329E+00,0.4335E+00,
     #     0.4342E+00,0.4348E+00,0.4355E+00,0.4361E+00,0.4368E+00,
     #     0.4374E+00,0.4381E+00,0.4387E+00,0.4394E+00,0.4400E+00,
```

```
     #    0.4406E+00,0.4413E+00,0.4419E+00,0.4426E+00,0.4432E+00,
     #    0.4438E+00,0.4445E+00,0.4451E+00,0.4458E+00,0.4464E+00/
          DATA (PCGTF3(I),I=801,850)/
     #    0.4470E+00,0.4474E+00,0.4478E+00,0.4481E+00,0.4485E+00,
     #    0.4489E+00,0.4493E+00,0.4497E+00,0.4500E+00,0.4504E+00,
     #    0.4508E+00,0.4512E+00,0.4516E+00,0.4519E+00,0.4523E+00,
     #    0.4527E+00,0.4531E+00,0.4535E+00,0.4538E+00,0.4542E+00,
     #    0.4546E+00,0.4550E+00,0.4554E+00,0.4558E+00,0.4561E+00,
     #    0.4565E+00,0.4569E+00,0.4573E+00,0.4577E+00,0.4581E+00,
     #    0.4584E+00,0.4588E+00,0.4592E+00,0.4596E+00,0.4600E+00,
     #    0.4604E+00,0.4608E+00,0.4612E+00,0.4615E+00,0.4619E+00,
     #    0.4623E+00,0.4627E+00,0.4631E+00,0.4635E+00,0.4639E+00,
     #    0.4643E+00,0.4647E+00,0.4650E+00,0.4654E+00,0.4658E+00/
          DATA (PCGTF3(I),I=851,900)/
     #    0.4662E+00,0.4666E+00,0.4670E+00,0.4674E+00,0.4678E+00,
     #    0.4682E+00,0.4686E+00,0.4690E+00,0.4694E+00,0.4698E+00,
     #    0.4702E+00,0.4706E+00,0.4710E+00,0.4713E+00,0.4717E+00,
     #    0.4721E+00,0.4725E+00,0.4729E+00,0.4733E+00,0.4737E+00,
     #    0.4741E+00,0.4745E+00,0.4749E+00,0.4753E+00,0.4757E+00,
     #    0.4761E+00,0.4765E+00,0.4769E+00,0.4773E+00,0.4777E+00,
     #    0.4781E+00,0.4785E+00,0.4789E+00,0.4793E+00,0.4798E+00,
     #    0.4802E+00,0.4806E+00,0.4810E+00,0.4814E+00,0.4818E+00,
     #    0.4822E+00,0.4826E+00,0.4830E+00,0.4834E+00,0.4838E+00,
     #    0.4842E+00,0.4846E+00,0.4850E+00,0.4854E+00,0.4858E+00/
          DATA (PCGTF3(I),I=901,950)/
     #    0.4863E+00,0.4867E+00,0.4871E+00,0.4875E+00,0.4879E+00,
     #    0.4883E+00,0.4887E+00,0.4891E+00,0.4895E+00,0.4899E+00,
     #    0.4904E+00,0.4908E+00,0.4912E+00,0.4916E+00,0.4920E+00,
     #    0.4924E+00,0.4928E+00,0.4933E+00,0.4937E+00,0.4941E+00,
     #    0.4945E+00,0.4949E+00,0.4953E+00,0.4958E+00,0.4962E+00,
     #    0.4966E+00,0.4970E+00,0.4974E+00,0.4978E+00,0.4983E+00,
     #    0.4987E+00,0.4991E+00,0.4995E+00,0.4999E+00,0.5004E+00,
     #    0.5008E+00,0.5012E+00,0.5016E+00,0.5020E+00,0.5025E+00,
     #    0.5029E+00,0.5033E+00,0.5037E+00,0.5042E+00,0.5046E+00,
     #    0.5050E+00,0.5054E+00,0.5059E+00,0.5063E+00,0.5067E+00/
          DATA (PCGTF3(I),I=951,Nrigs)/
     #    0.5071E+00,0.5076E+00,0.5080E+00,0.5084E+00,0.5088E+00,
     #    0.5093E+00,0.5097E+00,0.5101E+00,0.5106E+00,0.5110E+00,
     #    0.5114E+00,0.5119E+00,0.5123E+00,0.5127E+00,0.5131E+00,
     #    0.5136E+00,0.5140E+00,0.5144E+00,0.5149E+00,0.5153E+00,
     #    0.5157E+00,0.5162E+00,0.5166E+00,0.5170E+00,0.5175E+00,
     #    0.5179E+00,0.5184E+00,0.5188E+00,0.5192E+00,0.5197E+00,
     #    0.5201E+00,0.5205E+00,0.5210E+00,0.5214E+00,0.5219E+00,
     #    0.5223E+00,0.5227E+00,0.5232E+00,0.5236E+00,0.5241E+00,
     #    0.5245E+00,0.5249E+00,0.5254E+00,0.5258E+00,0.5263E+00,
     #    0.5267E+00,0.5271E+00,0.5276E+00,0.5280E+00,0.5285E+00,
     #    0.5289E+00/
          DATA (PCGTF4(I),I=1,50)/
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00/
          DATA (PCGTF4(I),I=51,100)/
     #    0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
```

```
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00/
      DATA (PCGTF4(I),I=101,150)/
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,
#     0.0000E+00,0.0000E+00,0.0000E+00,0.0000E+00,0.1998E-04,
#     0.3997E-04,0.5863E-04,0.7762E-04,0.9727E-04,0.1179E-03,
#     0.1399E-03,0.1636E-03,0.1896E-03,0.2185E-03,0.2510E-03,
#     0.2878E-03,0.3293E-03,0.3757E-03,0.4268E-03,0.4827E-03,
#     0.5433E-03,0.6086E-03,0.6784E-03,0.7529E-03,0.8318E-03/
      DATA (PCGTF4(I),I=151,200)/
#     0.9153E-03,0.1003E-02,0.1096E-02,0.1195E-02,0.1299E-02,
#     0.1410E-02,0.1528E-02,0.1653E-02,0.1787E-02,0.1929E-02,
#     0.2081E-02,0.2242E-02,0.2413E-02,0.2595E-02,0.2788E-02,
#     0.2991E-02,0.3205E-02,0.3428E-02,0.3660E-02,0.3901E-02,
#     0.4151E-02,0.4409E-02,0.4675E-02,0.4948E-02,0.5229E-02,
#     0.5516E-02,0.5809E-02,0.6109E-02,0.6416E-02,0.6730E-02,
#     0.7052E-02,0.7381E-02,0.7719E-02,0.8064E-02,0.8419E-02,
#     0.8782E-02,0.9154E-02,0.9536E-02,0.9928E-02,0.1033E-01,
#     0.1074E-01,0.1116E-01,0.1160E-01,0.1204E-01,0.1250E-01,
#     0.1296E-01,0.1344E-01,0.1394E-01,0.1444E-01,0.1496E-01/
      DATA (PCGTF4(I),I=201,250)/
#     0.1549E-01,0.1603E-01,0.1659E-01,0.1715E-01,0.1773E-01,
#     0.1832E-01,0.1892E-01,0.1952E-01,0.2013E-01,0.2074E-01,
#     0.2136E-01,0.2198E-01,0.2261E-01,0.2323E-01,0.2386E-01,
#     0.2448E-01,0.2510E-01,0.2572E-01,0.2634E-01,0.2695E-01,
#     0.2755E-01,0.2815E-01,0.2874E-01,0.2932E-01,0.2990E-01,
#     0.3046E-01,0.3100E-01,0.3154E-01,0.3207E-01,0.3259E-01,
#     0.3310E-01,0.3360E-01,0.3410E-01,0.3459E-01,0.3508E-01,
#     0.3556E-01,0.3604E-01,0.3652E-01,0.3700E-01,0.3748E-01,
#     0.3796E-01,0.3844E-01,0.3893E-01,0.3942E-01,0.3992E-01,
#     0.4042E-01,0.4093E-01,0.4145E-01,0.4197E-01,0.4251E-01/
      DATA (PCGTF4(I),I=251,300)/
#     0.4306E-01,0.4362E-01,0.4419E-01,0.4477E-01,0.4536E-01,
#     0.4596E-01,0.4658E-01,0.4720E-01,0.4783E-01,0.4847E-01,
#     0.4912E-01,0.4978E-01,0.5044E-01,0.5111E-01,0.5179E-01,
#     0.5248E-01,0.5317E-01,0.5386E-01,0.5457E-01,0.5527E-01,
#     0.5598E-01,0.5670E-01,0.5742E-01,0.5814E-01,0.5886E-01,
#     0.5959E-01,0.6032E-01,0.6105E-01,0.6178E-01,0.6251E-01,
#     0.6324E-01,0.6397E-01,0.6470E-01,0.6543E-01,0.6616E-01,
#     0.6689E-01,0.6762E-01,0.6834E-01,0.6906E-01,0.6978E-01,
#     0.7049E-01,0.7120E-01,0.7191E-01,0.7261E-01,0.7331E-01,
#     0.7399E-01,0.7468E-01,0.7536E-01,0.7603E-01,0.7669E-01/
      DATA (PCGTF4(I),I=301,350)/
#     0.7735E-01,0.7799E-01,0.7863E-01,0.7927E-01,0.7989E-01,
#     0.8051E-01,0.8112E-01,0.8173E-01,0.8233E-01,0.8293E-01,
#     0.8352E-01,0.8411E-01,0.8469E-01,0.8527E-01,0.8585E-01,
#     0.8642E-01,0.8699E-01,0.8756E-01,0.8813E-01,0.8869E-01,
#     0.8926E-01,0.8982E-01,0.9038E-01,0.9095E-01,0.9151E-01,
#     0.9208E-01,0.9264E-01,0.9321E-01,0.9378E-01,0.9435E-01,
```

```
     #     0.9492E-01,0.9550E-01,0.9607E-01,0.9666E-01,0.9724E-01,
     #     0.9784E-01,0.9843E-01,0.9903E-01,0.9964E-01,0.1003E+00,
     #     0.1009E+00,0.1015E+00,0.1021E+00,0.1028E+00,0.1034E+00,
     #     0.1041E+00,0.1047E+00,0.1054E+00,0.1061E+00,0.1068E+00/
           DATA (PCGTF4(I),I=351,400)/
     #     0.1075E+00,0.1082E+00,0.1089E+00,0.1096E+00,0.1104E+00,
     #     0.1111E+00,0.1119E+00,0.1126E+00,0.1134E+00,0.1142E+00,
     #     0.1150E+00,0.1158E+00,0.1166E+00,0.1174E+00,0.1182E+00,
     #     0.1191E+00,0.1199E+00,0.1208E+00,0.1216E+00,0.1225E+00,
     #     0.1233E+00,0.1242E+00,0.1251E+00,0.1260E+00,0.1268E+00,
     #     0.1277E+00,0.1286E+00,0.1295E+00,0.1304E+00,0.1314E+00,
     #     0.1323E+00,0.1332E+00,0.1341E+00,0.1350E+00,0.1360E+00,
     #     0.1369E+00,0.1379E+00,0.1388E+00,0.1397E+00,0.1407E+00,
     #     0.1416E+00,0.1426E+00,0.1435E+00,0.1445E+00,0.1454E+00,
     #     0.1464E+00,0.1474E+00,0.1483E+00,0.1493E+00,0.1502E+00/
           DATA (PCGTF4(I),I=401,450)/
     #     0.1512E+00,0.1522E+00,0.1531E+00,0.1541E+00,0.1550E+00,
     #     0.1560E+00,0.1570E+00,0.1579E+00,0.1589E+00,0.1598E+00,
     #     0.1608E+00,0.1618E+00,0.1627E+00,0.1637E+00,0.1647E+00,
     #     0.1656E+00,0.1666E+00,0.1675E+00,0.1685E+00,0.1695E+00,
     #     0.1704E+00,0.1714E+00,0.1723E+00,0.1733E+00,0.1743E+00,
     #     0.1752E+00,0.1762E+00,0.1771E+00,0.1781E+00,0.1791E+00,
     #     0.1800E+00,0.1810E+00,0.1819E+00,0.1829E+00,0.1838E+00,
     #     0.1848E+00,0.1858E+00,0.1867E+00,0.1877E+00,0.1886E+00,
     #     0.1896E+00,0.1905E+00,0.1915E+00,0.1925E+00,0.1934E+00,
     #     0.1944E+00,0.1953E+00,0.1963E+00,0.1972E+00,0.1982E+00/
           DATA (PCGTF4(I),I=451,500)/
     #     0.1991E+00,0.2001E+00,0.2010E+00,0.2020E+00,0.2029E+00,
     #     0.2039E+00,0.2048E+00,0.2058E+00,0.2067E+00,0.2077E+00,
     #     0.2086E+00,0.2096E+00,0.2105E+00,0.2115E+00,0.2124E+00,
     #     0.2134E+00,0.2143E+00,0.2153E+00,0.2162E+00,0.2172E+00,
     #     0.2181E+00,0.2191E+00,0.2200E+00,0.2209E+00,0.2219E+00,
     #     0.2228E+00,0.2238E+00,0.2247E+00,0.2256E+00,0.2266E+00,
     #     0.2275E+00,0.2285E+00,0.2294E+00,0.2303E+00,0.2313E+00,
     #     0.2322E+00,0.2331E+00,0.2341E+00,0.2350E+00,0.2359E+00,
     #     0.2369E+00,0.2378E+00,0.2387E+00,0.2396E+00,0.2406E+00,
     #     0.2415E+00,0.2424E+00,0.2434E+00,0.2443E+00,0.2452E+00/
           DATA (PCGTF4(I),I=501,550)/
     #     0.2461E+00,0.2471E+00,0.2480E+00,0.2489E+00,0.2498E+00,
     #     0.2507E+00,0.2517E+00,0.2526E+00,0.2535E+00,0.2544E+00,
     #     0.2553E+00,0.2562E+00,0.2571E+00,0.2581E+00,0.2590E+00,
     #     0.2599E+00,0.2608E+00,0.2617E+00,0.2626E+00,0.2635E+00,
     #     0.2644E+00,0.2653E+00,0.2662E+00,0.2671E+00,0.2680E+00,
     #     0.2689E+00,0.2698E+00,0.2707E+00,0.2716E+00,0.2725E+00,
     #     0.2734E+00,0.2743E+00,0.2752E+00,0.2760E+00,0.2769E+00,
     #     0.2778E+00,0.2787E+00,0.2796E+00,0.2805E+00,0.2813E+00,
     #     0.2822E+00,0.2831E+00,0.2840E+00,0.2848E+00,0.2857E+00,
     #     0.2866E+00,0.2875E+00,0.2883E+00,0.2892E+00,0.2901E+00/
           DATA (PCGTF4(I),I=551,600)/
     #     0.2909E+00,0.2918E+00,0.2926E+00,0.2935E+00,0.2944E+00,
     #     0.2952E+00,0.2961E+00,0.2969E+00,0.2978E+00,0.2986E+00,
     #     0.2995E+00,0.3003E+00,0.3012E+00,0.3020E+00,0.3028E+00,
     #     0.3037E+00,0.3045E+00,0.3053E+00,0.3062E+00,0.3070E+00,
     #     0.3078E+00,0.3087E+00,0.3095E+00,0.3103E+00,0.3111E+00,
     #     0.3120E+00,0.3128E+00,0.3136E+00,0.3144E+00,0.3152E+00,
     #     0.3160E+00,0.3168E+00,0.3176E+00,0.3184E+00,0.3192E+00,
     #     0.3200E+00,0.3208E+00,0.3216E+00,0.3224E+00,0.3232E+00,
     #     0.3240E+00,0.3248E+00,0.3256E+00,0.3263E+00,0.3271E+00,
     #     0.3279E+00,0.3287E+00,0.3294E+00,0.3302E+00,0.3310E+00/
           DATA (PCGTF4(I),I=601,650)/
```

```
#    0.3318E+00,0.3325E+00,0.3333E+00,0.3340E+00,0.3348E+00,
#    0.3355E+00,0.3363E+00,0.3370E+00,0.3378E+00,0.3385E+00,
#    0.3393E+00,0.3400E+00,0.3408E+00,0.3415E+00,0.3422E+00,
#    0.3430E+00,0.3437E+00,0.3444E+00,0.3451E+00,0.3459E+00,
#    0.3466E+00,0.3473E+00,0.3480E+00,0.3487E+00,0.3494E+00,
#    0.3502E+00,0.3509E+00,0.3516E+00,0.3523E+00,0.3530E+00,
#    0.3537E+00,0.3544E+00,0.3551E+00,0.3558E+00,0.3565E+00,
#    0.3572E+00,0.3578E+00,0.3585E+00,0.3592E+00,0.3599E+00,
#    0.3606E+00,0.3613E+00,0.3619E+00,0.3626E+00,0.3633E+00,
#    0.3640E+00,0.3646E+00,0.3653E+00,0.3660E+00,0.3666E+00/
     DATA (PCGTF4(I),I=651,700)/
#    0.3673E+00,0.3680E+00,0.3686E+00,0.3693E+00,0.3699E+00,
#    0.3706E+00,0.3712E+00,0.3719E+00,0.3725E+00,0.3732E+00,
#    0.3738E+00,0.3745E+00,0.3751E+00,0.3758E+00,0.3764E+00,
#    0.3771E+00,0.3777E+00,0.3783E+00,0.3790E+00,0.3796E+00,
#    0.3802E+00,0.3809E+00,0.3815E+00,0.3821E+00,0.3827E+00,
#    0.3834E+00,0.3840E+00,0.3846E+00,0.3852E+00,0.3858E+00,
#    0.3865E+00,0.3871E+00,0.3877E+00,0.3883E+00,0.3889E+00,
#    0.3895E+00,0.3901E+00,0.3907E+00,0.3914E+00,0.3920E+00,
#    0.3926E+00,0.3932E+00,0.3938E+00,0.3944E+00,0.3950E+00,
#    0.3956E+00,0.3962E+00,0.3968E+00,0.3974E+00,0.3980E+00/
     DATA (PCGTF4(I),I=701,750)/
#    0.3985E+00,0.3991E+00,0.3997E+00,0.4003E+00,0.4009E+00,
#    0.4015E+00,0.4021E+00,0.4027E+00,0.4032E+00,0.4038E+00,
#    0.4044E+00,0.4050E+00,0.4056E+00,0.4062E+00,0.4067E+00,
#    0.4073E+00,0.4079E+00,0.4085E+00,0.4090E+00,0.4096E+00,
#    0.4102E+00,0.4108E+00,0.4113E+00,0.4119E+00,0.4125E+00,
#    0.4130E+00,0.4136E+00,0.4142E+00,0.4147E+00,0.4153E+00,
#    0.4159E+00,0.4164E+00,0.4170E+00,0.4176E+00,0.4181E+00,
#    0.4187E+00,0.4193E+00,0.4198E+00,0.4204E+00,0.4209E+00,
#    0.4215E+00,0.4221E+00,0.4226E+00,0.4232E+00,0.4237E+00,
#    0.4243E+00,0.4248E+00,0.4254E+00,0.4259E+00,0.4265E+00/
     DATA (PCGTF4(I),I=751,800)/
#    0.4270E+00,0.4276E+00,0.4282E+00,0.4287E+00,0.4293E+00,
#    0.4298E+00,0.4304E+00,0.4309E+00,0.4315E+00,0.4320E+00,
#    0.4326E+00,0.4331E+00,0.4337E+00,0.4342E+00,0.4348E+00,
#    0.4353E+00,0.4358E+00,0.4364E+00,0.4369E+00,0.4375E+00,
#    0.4380E+00,0.4386E+00,0.4391E+00,0.4397E+00,0.4402E+00,
#    0.4408E+00,0.4413E+00,0.4419E+00,0.4424E+00,0.4429E+00,
#    0.4435E+00,0.4440E+00,0.4446E+00,0.4451E+00,0.4457E+00,
#    0.4462E+00,0.4467E+00,0.4473E+00,0.4478E+00,0.4484E+00,
#    0.4489E+00,0.4495E+00,0.4500E+00,0.4506E+00,0.4511E+00,
#    0.4516E+00,0.4522E+00,0.4527E+00,0.4533E+00,0.4538E+00/
     DATA (PCGTF4(I),I=801,850)/
#    0.4544E+00,0.4547E+00,0.4551E+00,0.4554E+00,0.4558E+00,
#    0.4561E+00,0.4565E+00,0.4568E+00,0.4571E+00,0.4575E+00,
#    0.4578E+00,0.4582E+00,0.4585E+00,0.4589E+00,0.4592E+00,
#    0.4596E+00,0.4599E+00,0.4603E+00,0.4606E+00,0.4610E+00,
#    0.4614E+00,0.4617E+00,0.4621E+00,0.4624E+00,0.4628E+00,
#    0.4631E+00,0.4635E+00,0.4638E+00,0.4642E+00,0.4645E+00,
#    0.4649E+00,0.4652E+00,0.4656E+00,0.4659E+00,0.4663E+00,
#    0.4667E+00,0.4670E+00,0.4674E+00,0.4677E+00,0.4681E+00,
#    0.4684E+00,0.4688E+00,0.4692E+00,0.4695E+00,0.4699E+00,
#    0.4702E+00,0.4706E+00,0.4709E+00,0.4713E+00,0.4717E+00/
     DATA (PCGTF4(I),I=851,900)/
#    0.4720E+00,0.4724E+00,0.4727E+00,0.4731E+00,0.4735E+00,
#    0.4738E+00,0.4742E+00,0.4746E+00,0.4749E+00,0.4753E+00,
#    0.4756E+00,0.4760E+00,0.4764E+00,0.4767E+00,0.4771E+00,
#    0.4775E+00,0.4778E+00,0.4782E+00,0.4785E+00,0.4789E+00,
#    0.4793E+00,0.4796E+00,0.4800E+00,0.4804E+00,0.4807E+00,
```

```
#    0.4811E+00,0.4815E+00,0.4818E+00,0.4822E+00,0.4826E+00,
#    0.4829E+00,0.4833E+00,0.4837E+00,0.4841E+00,0.4844E+00,
#    0.4848E+00,0.4852E+00,0.4855E+00,0.4859E+00,0.4863E+00,
#    0.4866E+00,0.4870E+00,0.4874E+00,0.4878E+00,0.4881E+00,
#    0.4885E+00,0.4889E+00,0.4892E+00,0.4896E+00,0.4900E+00/
     DATA (PCGTF4(I),I=901,950)/
#    0.4904E+00,0.4907E+00,0.4911E+00,0.4915E+00,0.4919E+00,
#    0.4922E+00,0.4926E+00,0.4930E+00,0.4934E+00,0.4937E+00,
#    0.4941E+00,0.4945E+00,0.4949E+00,0.4953E+00,0.4956E+00,
#    0.4960E+00,0.4964E+00,0.4968E+00,0.4971E+00,0.4975E+00,
#    0.4979E+00,0.4983E+00,0.4987E+00,0.4990E+00,0.4994E+00,
#    0.4998E+00,0.5002E+00,0.5006E+00,0.5009E+00,0.5013E+00,
#    0.5017E+00,0.5021E+00,0.5025E+00,0.5029E+00,0.5032E+00,
#    0.5036E+00,0.5040E+00,0.5044E+00,0.5048E+00,0.5052E+00,
#    0.5056E+00,0.5059E+00,0.5063E+00,0.5067E+00,0.5071E+00,
#    0.5075E+00,0.5079E+00,0.5083E+00,0.5086E+00,0.5090E+00/
     DATA (PCGTF4(I),I=951,Nrigs)/
#    0.5094E+00,0.5098E+00,0.5102E+00,0.5106E+00,0.5110E+00,
#    0.5114E+00,0.5118E+00,0.5121E+00,0.5125E+00,0.5129E+00,
#    0.5133E+00,0.5137E+00,0.5141E+00,0.5145E+00,0.5149E+00,
#    0.5153E+00,0.5157E+00,0.5161E+00,0.5165E+00,0.5169E+00,
#    0.5172E+00,0.5176E+00,0.5180E+00,0.5184E+00,0.5188E+00,
#    0.5192E+00,0.5196E+00,0.5200E+00,0.5204E+00,0.5208E+00,
#    0.5212E+00,0.5216E+00,0.5220E+00,0.5224E+00,0.5228E+00,
#    0.5232E+00,0.5236E+00,0.5240E+00,0.5244E+00,0.5248E+00,
#    0.5252E+00,0.5256E+00,0.5260E+00,0.5264E+00,0.5268E+00,
#    0.5272E+00,0.5276E+00,0.5280E+00,0.5284E+00,0.5288E+00,
#    0.5292E+00/

      DO I=1,Nrigs
       RigBins(I)=RIGPC(I)
      ENDDO

    RETURN
    END
```

```
      SUBROUTINE ConvertTime(time,UTtimeInit,UTtime,Zlon,Period,
     #                      TimeLocal)

C     Calculate Universal Time and Local Time.  Note that Zlon
C     and period are not both required.  Used as a consistency check
C     here.

      IMPLICIT NONE
      REAL Time,UTtimeInit,UTtime,Zlon,Period,TimeLocal
      REAL secsperday
      PARAMETER (secsperday=86400.0)
      Integer Ndays,Ndaysloc
      REAL TimeLoc

C-----------------------------------------------------------------------

      UTtime=UTtimeInit+time
      Ndays=INT(UTtime/secsperday)
      UTtime=UTtime - Ndays*secsperday

      TimeLocal=UTtime+secsperday*Zlon/360.0
      IF (TimeLocal .GE. secsperday) TimeLocal=TimeLocal-secsperday

      RETURN
      END


C-----------------------------------------------------------------------

      SUBROUTINE OutputTransFcn(RigBins,TransFunc,GtransFile,OrbIncl,
     #                 Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp,
     #                 Zenith,Azimuth,UTtimeInit,Stormy,Shadow,
     #                 PreCalcGTFs,IPreCalc,Year,XLbounds,
     #                 ILbins,IprogNo)

      IMPLICIT NONE
C
C     Format of header changed by AJT 8-21-96.
C     File names for L-bin results changed from .BN* to .GT* by AJT 11-6-96

C     IF one L-bin specified with L_min > 0, make file extension .GT1
C     instead of .GTF.  11-21-96, PRB.

      REAL OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp

C     These variables are fixed at present and thus not included
C     in header output.
      REAL Zenith,Azimuth,UTtimeInit

C     Shadow is not checked in determining output, since all GTF
C     calculations included in GEOMAG96 are omnidirectional averages.

      LOGICAL Shadow,Stormy,PreCalcGTFs,StormyPreCalc

      INTEGER IPreCalc

C     Name of output file, thus not included in output file header.
      CHARACTER*80 GtransFile,TempFile

      INTEGER I,Nrigs,NLvals,L,ILbins
```

```
      PARAMETER (Nrigs=1001,NLvals=10)

      REAL TransFunc(Nrigs,NLvals),RigBins(Nrigs)

      REAL XLbounds(NLvals),XLinfinite,Year
      PARAMETER (XLinfinite=1.0E+06)

      INTEGER ICREME96vno,IProgNo,IPreCalcOutput,IstormOutput
      INTEGER NHEADER,STAT,CREME96_OPEN
      DATA NHEADER/3/
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME


      CHARACTER*5 FEXT(10)

      DATA FEXT/'.GT1','.GT2','.GT3','.GT4','.GT5','.GT6','.GT7',
     &          '.GT8','.GT9','.GTX'/

C-----------------------------------------------------------------------

      CALL GET_CREME96_VERSION(ICREME96vno)
      IPreCalcOutput=0

      IF (PreCalcGTFs) THEN
        StormyPreCalc=.FALSE.   !Local variable for header output file.
        IPreCalcOutput=1        !Local variable for header output file.
        IF ( (IpreCalc .EQ. 1) .OR. (IpreCalc .EQ. 3) )
     &       StormyPreCalc=.TRUE.
      ENDIF

      IStormOutput=0
      IF (StormyPreCalc .OR. Stormy) IStormOutput=1

C     ILbins = 0 & ILbins = 1 from input routine are treated
C     as ILbins = 1 for output, since they are stored in the
C     same location in the array.

      IF (ILbins .EQ. 1 .AND. XLBOUNDS(1) .EQ. 0.0) THEN
c          OPEN(UNIT=16,STATUS='NEW',FILE='USER:'//GtransFile)
          stat = creme96_open(gtransfile,'user',16,'new')
          CALL DATE(CREATION_DATE)
          CALL TIME(CREATION_TIME)

          WRITE(16,403) NHEADER,GTRANSFILE(1:70),
     #                  ICREME96vno,IProgno

          WRITE(16,992) ICREME96vno,CREATION_DATE,CREATION_TIME
          WRITE(16,404) OrbIncl,Apogee,Perigee,AscNodeLong,
     #                  AscNodeDisp,PerigDisp
          WRITE(16,405) IStormoutput,IPrecalcOutput,Year,
     #                  XLbounds(ILbins),XLbounds(Ilbins+1)

          DO I=1,Nrigs
            WRITE(16,410)RigBins(I),TransFunc(I,ILbins)
          ENDDO
          CLOSE(16)

      ELSE
```

```
            DO L=1,ILbins
              TEMPFILE=Gtransfile(1:index(gtransfile,' ')-1)//FEXT(L)
c              OPEN(UNIT=16,STATUS='NEW',FILE='USER:'//GtransFile//FEXT(L))
              stat = creme96_open(tempfile,'user',16,'new')
              CALL DATE(CREATION_DATE)
              CALL TIME(CREATION_TIME)
              WRITE(16,403) NHEADER,TEMPFILE(1:70),
     #                      ICREME96vno,IProgno
              WRITE(16,992) ICREME96vno,CREATION_DATE,CREATION_TIME

              WRITE(16,404) OrbIncl,Apogee,Perigee,AscNodeLong,
     #                      AscNodeDisp,PerigDisp

              IF (L .LT. NLvals) THEN
                  WRITE(16,405) IStormoutput,IPrecalcOutput,Year,
     #                      XLbounds(L),XLbounds(L+1)
              ELSE
                  WRITE(16,405) IStormoutput,IPrecalcOutput,Year,
     #                      XLbounds(L),XLinfinite
              ENDIF


              DO I=1,Nrigs
                WRITE(16,410)RigBins(I),TransFunc(I,L)
              ENDDO

              CLOSE(16)
            ENDDO

          ENDIF

          RETURN
 403      FORMAT(I3,1x,A70,I4,1x,I1)
 404      FORMAT(1x,'%Incl = ',F7.3,' deg  Apo = ',E10.4,
     #              ' Peri = ',E10.4,' km',1x,3(F6.2,1x))
     #
 405      FORMAT(1x,'%ISTORM =',I2,' IPRECALC =',I2,
     #              '  Grid Epoch = ',F6.1,'    L Bin: ',2(E10.4,1X))

 991      FORMAT(1x,A79)
 992      FORMAT(1x,'%Created by CREME96:GTRANS_DRIVER Version ',I4,
     &              ' on ',A9,' at ',A8)

 410      FORMAT(5X,F6.3,5X,E10.4)

          END   !OutputTransFunc routine


C-----------------------------------------------------------------------

          SUBROUTINE GetPreCalcGTF(IPreCalc,RigBins,TransFunc)
C
          IMPLICIT NONE
          INTEGER I,IPreCalc,Nrigs,NLvals
          PARAMETER(Nrigs=1001,NLvals=10)

          LOGICAL PreCalcInit

C         contain output rigidity vs. transmission function
          REAL RigBins(Nrigs),TransFunc(Nrigs,NLvals)
```

```fortran
      REAL RIGPC(Nrigs)
      REAL PCGTF1(Nrigs),PCGTF2(Nrigs),PCGTF3(Nrigs),PCGTF4(Nrigs)

      COMMON/PreCalcCMN/PCGTF1,PCGTF2,PCGTF3,PCGTF4

      DATA PreCalcInit/.FALSE./

C-----------------------------------------------------------------------


C     Initialize pre-calculated GTFs

      IF (.NOT. PreCalcInit) THEN
        CALL InitPreCalcs(RigBins)
        PreCalcInit=.TRUE.
      ENDIF

C     Set these each time, providing capability to change
C     if input is so structured

      IF (IpreCalc .EQ. 0) THEN
        DO I=1,Nrigs
         TransFunc(I,1)=PCGTF1(I)
        ENDDO
      ENDIF

      IF (IpreCalc .EQ. 1) THEN
        DO I=1,Nrigs
         TransFunc(I,1)=PCGTF2(I)
        ENDDO
      ENDIF

      IF (IpreCalc .EQ. 2) THEN
        DO I=1,Nrigs
         TransFunc(I,1)=PCGTF3(I)
        ENDDO
      ENDIF

      IF (IpreCalc .EQ. 3) THEN
        DO I=1,Nrigs
         TransFunc(I,1)=PCGTF4(I)
        ENDDO
      ENDIF

      RETURN
      END

C-----------------------------------------------------------------------

      SUBROUTINE CALCULATE_TRANS_FUNC(ISTEP,MAT,CF,NperLbin,T)
C
C     Modified to allow the transmission function to be nonzero for
C     C=0 bin (0.0-0.2 GV at present) , 1-29-96, PRB.

C     Modified MAT to be real, in order to handle geometric shadowing
C     for non circular orbits in a consistent manner

C     Removed check on NstepSum being equal to ISTEP, since the minimum
C     and maximum L-values of the specified bins do not have to
```

```
C       include all L-values in the orbit.  11-21-96, PRB.
C
        IMPLICIT NONE

        INTEGER ISTEP,J,Nrigs,NLvals,NstepSum,L
        PARAMETER(Nrigs=1001,NLvals=10)

        INTEGER NperLbin(NLvals)
        REAL MAT(Nrigs,NLvals)
        REAL T(Nrigs,NLvals),CF(Nrigs),CMAT(NLvals)

C-----------------------------------------------------------------------

C


        NstepSum=NperLbin(1)

        DO L=1,NLvals
          CMAT(L)=0.
          IF (L .GE. 2) NstepSum=Nstepsum+NperLbin(L)
        ENDDO
C
        DO L=1,NLvals
          IF (NperLbin(L) .GT. 0) THEN

            DO J=1,Nrigs
C
C         Convert the histogram to transmission.
C
              CMAT(L)=MAT(J,L)/FLOAT(NperLbin(L))+CMAT(L)
              IF (L .EQ. 1) CF(J)=FLOAT(J-1)*0.02
              T(J,L)=CMAT(L)

            ENDDO

          ENDIF
        ENDDO
C
        RETURN

        END

C-----------------------------------------------------------------------

        SUBROUTINE NYMMIK(ReffGrid,TimeLocal,DeltaNymmik)

C       PURPOSE: To calculate Nymmik parameterization of Cutoff for
C       Rigidities below 1 GV, given the IGRF model.  This attempts
C       to account for the quiet external magnetospheric fields for
C       high latitudes.  The 1980 Shea & Smart 5 deg. by 5 deg. grid is
C       expected for the IGRF model.  Uses standard local time as
C       geomagnetic local time


        IMPLICIT NONE
        REAL ReffGrid,ReffNymmik,DeltaNymmik
        REAL TimeLocal,HoursLT
        REAL PI
```

```
C-----------------------------------------------------------------------

         PI=ACOS(-1.0)

         HoursLT= TimeLocal/3600.

C        Reset correction to zero in case grid is zero

         DeltaNymmik = 0.

C         ReffNymmik=ReffGrid   !handled in Geomag.

         IF (ReffGrid .LE. 1.0 .AND. ReffGrid .GT. 0.)THEN
           DeltaNymmik=1.42*(0.67/ReffGrid)**(1.1+
     >                  1.62*COS(2*PI/24*(HoursLT+0.6)))

           DeltaNymmik=DeltaNymmik/EXP(1.72*(ReffGrid/0.67)**2*
     >                (1+0.66*SIN(2*PI/12*(HoursLT+2.0))))
C
C          Actual correction performed in SUBROUTINE Geomag
C           ReffNymmik=ReffGrid/(1+DeltaNymmik)

         ENDIF

         RETURN
         END

C-----------------------------------------------------------------------

         SUBROUTINE Orbit(n,time,zlon,zlat,radius,alta,altp,a1,a2,a3,xi)

C        THIS SUBROUTINE USED TO ACCEPT INPUT CONCERNING SATELLITE
C        ORBITS AND CALCULATES THEIR GEOGRAPHICAL LOCATION.
C        N=0:   disabled here, since the driver program now establishes
C               all initial values.
C        N=1:   Initialize data on orbit (complete mode).
C        N=2:   CALCULATE ORBIT AS A FUNCTION OF TIME.
C        Clearly data must be input before computations.
C        On data input, time returns the orbital period.
C        During orbit calculations, time is an input variable.

C        The following data should be passed into this Subroutine.
C        Re is now contained in a DATA statement.  Only E (the eccentricity)
C        is calculated in this version.
C
C        ALTA=Orbital altitude at apogee (kilometers).
C        ALTP=Orbital altitude at perigee (kilometers).
C        RE=Radius of Earth (kilometers).
C        E=Orbit eccentricity.
C        a1=Orbital inclination (degrees).
C        A2=Initial longitude of ascending node (degrees).
C        A3=Initial displacement from ascending node (degrees).
C        XI=Displacement of perigee from ascending node (degrees).
C
         IMPLICIT REAL (A-H,O-Z)

         INTEGER N
         REAL pi,alta,altp,Re,e,a1,a2,a3,xi,w1,rmaj,w2,fact
         REAL tho,pho,psi,xio
         REAL time,zlon,zlat,radius
```

```
      INTEGER I1014STEPS
      REAL DELMAX
C
      DATA Re/6371.2/

C-----------------------------------------------------------------------


      pi=4.0*ATAN(1.0)      !use ARCTAN to calculate PI

      IF (N.EQ.2) GO TO 1000
      E=(ALTA-ALTP)/(ALTA+ALTP+2.*RE)
      IF (E.LT..00001) E=0.

C
C     W1=ANGULAR VELOCITY OF EARTH (RADIANS/SEC).
C     RMAJ=SEMI-MAJOR AXIS (KILOMETERS).
C     W2=MEAN ORBITAL ANGULAR VELOCITY (RADIANS/SECOND).
C     TIME=ORBITAL PERIOD (SECONDS).
C     FACT=A USEFUL FACTOR.
C
      W1=7.27E-5
      RMAJ=(ALTP+RE)/(1.-E)
      W2=1.24E-3*(RE/RMAJ)**1.5
      TIME=2.*PI/W2
      FACT=SQRT((1.+E)/(1.-E))
C
C     DEFINE MORE USEFUL ANGLES.
C
      THO=PI*A1/180.
      PHO=PI*(A2-90.)/180.
      PSI=PI*A3/180.
      XIO=PI*XI/180.

      RETURN    !Finished initializing orbital variables
 1000 CONTINUE
C
C     COMPUTE SATELLITE POSITION.
C
C     QM=MEAN ANOMALY.
C
      IF (E.NE.0.) GOTO 1009
        QM=W2*TIME+PSI
      GOTO 1010
 1009 CONTINUE
        YSS=(PSI-XIO)/2.
        QE0=2.*ATAN2(SIN(YSS),FACT*COS(YSS))
        QM0=QE0-E*SIN(QE0)
        QM=W2*TIME-QM0
 1010 CONTINUE
C
C     QE=ECCENTRIC ANOMALY.
C
      QE=QM
      DEL=1.

      I1014STEPS=0
      DELMAX=.0001
 1014 CONTINUE
```

```
          I1014STEPS=I1014STEPS+1
          QTEMP=QE
          QE=QM+E*SIN(QE)
          DEL=QE-QTEMP

        IF (I1014STEPS .GE. 50) DELMAX=0.002
        IF (I1014STEPS .GE. 100) DELMAX=0.005
        IF (I1014STEPS .GE. 200) DELMAX=0.01
        IF (I1014STEPS .GE. 500) DELMAX=0.02
        IF (I1014STEPS .GE. 1000) DELMAX=0.05

        IF (ABS(DEL).GT.DELMAX .AND. I1014STEPS .LT. 2000) GOTO 1014
C
C       QT=TRUE ANOMALY.
C
        IF (E.NE.0.) GOTO 1019
          QT=QE
        GOTO 1020
 1019   CONTINUE
          QECYC=INT(QE/2./PI)
          QERED=QE-2.*PI*QECYC
          AA1=FACT*SIN(QERED/2.)
          AA2=COS(QERED/2.)
          QTRED=2.*ATAN2(AA1,AA2)
          IF (QTRED.LT.0.) QTRED=QTRED+2.*PI
          QT=QTRED+2.*PI*QECYC+XIO
C
C       NOTE:  ANOMALY COMPUTATIONS ARE DONE FROM PERIGEE
C       WHILE ORBIT COMPUTATIONS BELOW ARE DONE FROM THE
C       ASCENDING NODE.  THE FACTOR OF XIO CORRECTS THIS.
C
 1020   CONTINUE
C
C       ZLAT=LATITUDE.
C
        R1=SIN(THO)*SIN(QT)
        THP=ACOS(R1)
        ZLAT=90.-180.*THP/PI
C
C       ZLON=LONGITUDE.
C
        RP=.5*(PI/2.+THO)
        RM=.5*(PI/2.-THO)
        RF=.5*(PI/2.-QT)
        IF (SIN(RF).NE.0.) GOTO 1029
          PHP=PI+PHO-W1*TIME
        GOTO 1030
 1029   CONTINUE
          S1=SIN(RM)*COS(RF)/SIN(RP)/SIN(RF)
          S2=COS(RM)*COS(RF)/COS(RP)/SIN(RF)
          SUM=ATAN(S1)+ATAN(S2)
          PHP=PHO-W1*TIME+SUM
 1030   CONTINUE
        IF (PHP.GE.0.) GOTO 1034
          PHP=PHP+2.*PI
        GOTO 1030
 1034   CONTINUE
        IF (PHP.LT.2.*PI) GOTO 1035
          PHP=PHP-2.*PI
        GOTO 1034
```

```
 1035   CONTINUE
        ZLON=180.*PHP/PI
C
C       RADIUS=ALTITUDE (KILOMETERS).
C
        RADIUS=RMAJ*(1.-E*COS(QE))-RE
        RETURN      !Actual orbital step computations
        END
```

```
      PROGRAM GtransDriver
C
C     This program calculates the transmission functions as proposed in
C     in the first year of the SEE work.  It is intended as a driver
C     program for subroutines which will be used in the overall program.
C
C     At present, it does not allow for repeated calls.  The overall
C     structure should allow repeated calls, but would need to be tested.
C
      IMPLICIT NONE

      INTEGER Nrigs,NLvals
      PARAMETER (Nrigs=1001,NLvals=10)

C     Rigidities expected in 0.02 GV steps & common for all L-bins

      REAL TransFunc(Nrigs,NLvals),RigBins(Nrigs),XLbounds(NLvals)

C     Parameters that are input or initialized in GTFDriverInput
C     These need to be passed to GEOMAG3.

      REAL OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp
      REAL Zenith,Azimuth,UTtimeInit,Year
      INTEGER IpreCalc,ILbins
      LOGICAL Shadow,Stormy,PreCalcGTFs
      CHARACTER*80 GtransFile

      INTEGER Iprogno
      DATA IprogNo/2/

C-----------------------------------------------------------------------

      CALL GTFDriverInput(OrbIncl,Apogee,Perigee,AscNodeLong,
     #       AscNodeDisp,PerigDisp,Zenith,Azimuth,UTtimeInit,Stormy,
     #       Shadow,PreCalcGTFs,IPreCalc,GtransFile,Year,XLbounds,
     #       ILbins)


      CALL Geomag96(OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,
     #        PerigDisp,Zenith,Azimuth,UTtimeInit,Stormy,Shadow,
     #        PreCalcGTFs,IPreCalc,RigBins,TransFunc,Year,XLbounds,
     #        ILbins)

c     For adding header information to output GTF file. Added July 1996.

      CALL OutputTransFcn(RigBins,TransFunc,GtransFile,OrbIncl,Apogee,
     #       Perigee,AscNodeLong,AscNodeDisp,PerigDisp,Zenith,Azimuth,
     #       UTtimeInit,Stormy,Shadow,PreCalcGTFs,IPreCalc,Year,
     #       XLbounds,ILbins,IprogNo)


      STOP
      END


C-----------------------------------------------------------------------

      SUBROUTINE GTFDriverInput(OrbIncl,Apogee,Perigee,AscNodeLong,
     #             AscNodeDisp,PerigDisp,Zenith,Azimuth,UTtimeInit,
     #             Stormy,Shadow,PreCalcGTFs,IPreCalc,GtransFile,
```

```
     #                Year,XLbounds,ILbinsum)

      IMPLICIT NONE

      REAL OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp
      REAL Zenith,Azimuth,UTtimeInit

C     Note that the eccentricity is calculated here to decide if
C     need to read PerigDisp.  The eccentricity is also recalculated
C     in the initialization CALL ORBIT(1,...) case found in SUBROUTINE GEOMAG
C     This makes the input driver independent of the actual computational
C     routines, so that it will be easier to modify and interface with other
C     space environment routines.

      REAL E,Re      !eccentricity and radius of Earth
      PARAMETER (Re=6371.2)
      LOGICAL Shadow,Stormy,PreCalcGTFs
      REAL ApPerSwitch

      INTEGER Istorm,Ishadow,IPreCalc
      INTEGER Itype,IGTFtype   !look-out directions & pre/non-pre calc. GTFs
      INTEGER IERR,IACCEPT
      DATA IERR/0/

      CHARACTER*80 GtransFile

      INTEGER NLvals,I,L,ILbinMax,ILbinsum
      PARAMETER (NLvals=10)

      REAL XLbounds(NLvals),XLinfinite,Year
      PARAMETER (XLinfinite=1.0E+06)

      REAL XLdummy

C-----------------------------------------------------------------------

      WRITE(6,1000)
      WRITE(6,1001)

      UTtimeInit=0.0         !start at 0 UT by default

C     Present averaging algorithms assume that zenith & azimuth
C     correspond to vertical incidence.

      Zenith=0.0
      Azimuth=0.0

C     initialize boundaries L-value bins

      XLbounds(1)=0.0
      Year=1980.0  !needed for L-value calculations

      DO L=2,NLvals
        XLbounds(L)=XLinfinite
      ENDDO

C     Check if user wants to use a pre-calculated GTF.  If so, read
C     specified option and return.
 9390 CONTINUE
```

```
      CALL RETRY_INPUT(IERR)
      WRITE(*,390)
      READ(*,*,ERR=9390,IOSTAT=IERR)IGTFtype
      PreCalcGTFs=.FALSE. !initialize not to use pre-calculated GTFs

C     Note that pre-calculated GTFs already include Earth shadow,
C     since solid Earth is included in the trajectory-tracing calculations.
C     In those cases, Shadow must be set to .FALSE. regardless of user
C     input.

      IF (IGTFtype .NE. 0) THEN
 9391   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,391)
        READ(*,*,ERR=9391,IOSTAT=IERR)IpreCalc
        PreCalcGTFs=.TRUE.
        Shadow=.FALSE.

C       Use quiet-time, 51.6 degrees as the default case

        IF (IpreCalc .LT. 0 .OR. IpreCalc .GT. 3) IpreCalc=0
 9427   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,427)
        READ(*,428,ERR=9427,IOSTAT=IERR)GtransFile
        CALL CHECK_OUTPUT_FILE(Gtransfile,IACCEPT)
        IF (IACCEPT.NE.0) GOTO 9427

C       For use in SUBROUTINE GTFHeaderOutput.  Added July 1996.

        IF (IpreCalc .EQ. 0 .OR. IpreCalc .EQ. 1) THEN
          OrbIncl=51.6
          Apogee=450.0
          Perigee=450.0
        ELSEIF (IpreCalc .EQ. 2 .OR. IpreCalc .EQ. 3) THEN
          OrbIncl=28.5
          Apogee=450.0
          Perigee=450.0
        ENDIF

C       The pre-calculated GTFs are not presently divided into L-bins
        ILbinsum=1

        RETURN

      ENDIF

C     Hardwire shadow to be TRUE

      Shadow=.TRUE.

C
C     Choose from the two original CREME options for the state of the
C     magnetosphere (quiet or stormy).  Note the stormy option applies on
C     top of the Nymmik correction for mid to high-latitudes.
C
 9412 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE (*,412)
      READ (*,*,ERR=9412,IOSTAT=IERR) Istorm
```

```
      Stormy=.FALSE.
      IF (Istorm  .EQ. 1) Stormy=.TRUE.


C     What is the altitude at apogee?
C
 9420 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(*,420)
      READ (*,*,ERR=9420,IOSTAT=IERR)  Apogee
C
C     WHAT IS THE ALTITUDE AT PERIGEE?
C
 9400 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(*,400)
      READ (*,*,ERR=9400,IOSTAT=IERR)  Perigee

C     allow the user to specify apogee and perigee in either order
C     instead of performing unintended calculation which sets eccentricity
C     to zero and using Perigee variable (actual apogee) to produce
C     a circular orbital altitude in ORBIT routine.

      IF (Perigee .GT. Apogee) THEN
         ApPerSwitch=Apogee
         Apogee=Perigee
         Perigee=ApPerSwitch
         WRITE(*,430)
      ENDIF

      E=(Apogee-Perigee)/(Apogee+Perigee+2.*Re)
      IF (E.LT..00001) E=0.
C
C     WHAT IS THE ORBITAL INCLINATION?
C
 9405 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(*,405)
      READ (*,*,ERR=9405,IOSTAT=IERR)  OrbIncl

C     Have Removed "FAST" option, i.e. must enter Ascending Node information
C
C     Retain these initializations in case want to hardwire ascending
C     node information at future time.

      AscNodeLong=0.
      AscNodeDisp=0.
      PerigDisp=0.

C     WHAT IS THE INITIAL LONGITUDE OF THE ASCENDING NODE?
C
      WRITE(*,409)

 9410 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(*,410)
      READ (*,*,ERR=9410,IOSTAT=IERR) AscNodeLong
C
C     WHAT IS THE INITIAL DISPLACEMENT FROM THE ASCENDING NODE?
C
```

```
 9415   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,415)
        READ (*,*,ERR=9415,IOSTAT=IERR)  AscNodeDisp

        IF (E.NE.0.) THEN       !Only read in XI if eccentricity is nonzero
C
C           What is the displacement of the perigee from the ascending node?
C
 9425    CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(*,425)
         READ (*,*,ERR=9425,IOSTAT=IERR)  PerigDisp

        ENDIF

        IF ( (AscNodeLong .NE. 0.0) .OR. (AscNodeDisp .NE. 0.0) .OR.
     &       (PerigDisp .NE. 0.0) ) WRITE(*,426)

        Itype = 1   !hardwire vertical incidence, applied with shadow.
C
 9450   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,450)
        READ (*,*,ERR=9450,IOSTAT=IERR)  ILbinMax

        IF (ILbinMax .LT. 0) ILbinMax=0

        IF (ILbinMax .GT. NLvals) THEN
           WRITE(*,456)
           ILbinMax = NLvals
        ENDIF

        IF (ILbinMax .GT. 0) THEN
          WRITE(*,451) ILbinMax
 9451       CONTINUE
            CALL RETRY_INPUT(IERR)
            READ (*,*,ERR=9451,IOSTAT=IERR) (XLbounds(L),L=1,IlbinMax)
            IF (ILbinMax .EQ. 1 .AND. XLBOUNDS(1) .EQ. 0.0) WRITE(*,458)
        ENDIF

        IF (XLbounds(1) .LT. 0.0) XLbounds(1)=0.0

C       Start DO loop at 1, so that ILbinMax=2 will be properly handled
C       This SUBROUTINE insists the L-values are in increasing order.
C       If this is not the case, all subsequent L-value bins will be
C       ignored.

        DO L=1,ILbinMax
          IF (XLbounds(L) .LT. XLbounds(1)) THEN
              WRITE(*,452) XLbounds(L),XLinfinite
              XLbounds(L)=XLinfinite
          ENDIF
          IF (L .GE. 2) THEN
            IF (XLbounds(L) .LE. XLbounds(L-1)) THEN
              WRITE(*,452) XLbounds(L),XLinfinite
              XLbounds(L)=XLinfinite
            ENDIF
          ENDIF
        ENDDO
```

```
      ILbinsum=1

      DO L=1,ILbinMax
        IF ( (L .GE. 2) .AND. (XLbounds(L) .LT. XLinfinite) )
     &        ILbinsum=ILbinsum+1
      ENDDO

      IF (ILbinMax .NE. ILbinsum .AND. ILbinMax .NE. 0) THEN
        WRITE(*,453)ILbinMax,ILbinsum
        ILbinMax=ILbinsum
      ENDIF

      IF ( ILbinMax .GT. 1 .OR. (ILbinMax .EQ. 1 .AND.
     &     XLBOUNDS(1) .GT. 0.0) ) THEN
9454    CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,454)
        READ (*,*,ERR=9454,IOSTAT=IERR) Year
      ENDIF

9428  CONTINUE
      CALL RETRY_INPUT(IERR)
      IF (ILbinMax .EQ. 0 .OR. (ILbinMax .EQ. 1 .AND.
     &     XLBOUNDS(1) .EQ. 0.0) ) THEN
        WRITE(*,427)
        READ(*,428,ERR=9428,IOSTAT=IERR)GtransFile
        CALL CHECK_OUTPUT_FILE(Gtransfile,IACCEPT)
        IF (IACCEPT.NE.0) GOTO 9428
      ELSE
        WRITE(*,455)ILbinMax
        READ(*,428,ERR=9428,IOSTAT=IERR)GtransFile
        CALL CHECK_OUTPUT_FILE(Gtransfile,IACCEPT)
        IF (IACCEPT.NE.0) GOTO 9428
        DO I=1,LEN(GtransFile)
          IF (GtransFile(I:I) .EQ. '.') THEN
              GtransFile=GtransFile(1:I-1)
          ENDIF
        ENDDO
      ENDIF

      RETURN


1000  FORMAT(1X,'GEOMAG96 Geomagnetic Transmission Function Model',/)
1001  FORMAT(' This program will calculate the omnidirectional',
     &        ' geomagnetic transmission',
     &      /,' function (GTF) to a',
     &        ' spacecraft orbiting inside the magnetosphere.  The',
     &      /,' calculated GTF is used by the CREME96',
     &        ' particle environment model.',
     &//,' NOTE: Before running this or any other CREME96 programs',
     &   ' please define three ',
     & /,' logicals: ',/,
     & /,4x,' CREME96       as the directory where CREME96 source',
     &        ' & executables reside.',
     & /,4x,' CR96TABLES    as the directory in which CREME96 data',
     &        ' tables reside.',
     & /,4x,' USER          as the directory in which output files',
     &        ' should be written.',
```

```
     & //,' Now begin specification of the GTF calculation: ',/)

C                      1         2         3         4         5
C             12345678901234567890123456789012345678901234567890
 390      FORMAT(1X,'Enter 1 in order to use a pre-calculated GTF for a',
     &           ' typical space shuttle or',
     &  /,3x,' space station orbit, ie., 28.5 deg or 51.6 deg & 450 km.'
     & //,1X,'Enter 0 to specify an arbitrary orbit: ',
     & //,3X,'[The pre-calculated GTFs are recommended if appropriate,',
     &       ' since these use a'
     &  /,3x,' better magnetic field model than used in the arbitrary',
     &       ' orbit option.]')

 391      FORMAT(1X,'Enter 0 for Space Station (51.6 deg., 450 km)',
     &       ' orbit (ISSA), quiet magnetosphere',
     &  /,7X,'1 for ISSA, stormy magnetosphere',
     &  /,7X,'2 for 28.5 deg. (450 km), quiet magnetosphere',
     &  /,7X,'3 for 28.5 deg. (450 km), stormy magnetosphere: ',
     &  //,3X,'[(a) For solar-quiet periods, the quiet magnetosphere',
     &        ' is typical.',
     &   /,3X,' (b) For solar energetic particles, the stormy ',
     &        'magnetosphere should also'
     &  /,8X,'be considered.]',
     & //,3X,'NOTE: the Worst Day in 22 years (see the CREME96',
     &       ' environment model)'
     &  /,3X,'included a stormy magnetosphere,',
     &       ' and thus a stormy option must be considered',
     &  /,3X,'with this Worst Day option.')

 412      FORMAT(/,1X,'Enter the magnetospheric field condition:  0 ',
     #   'for quiet; 1 for stormy:  ',
     &  //,3X,'[(a) For solar-quiet periods, the quiet magnetosphere',
     &        ' is typical.',
     &   /,3X,' (b) For solar energetic particles, the stormy ',
     &        'magnetosphere should also'
     &  /,8X,'be considered.]',
     & //,3X,'NOTE: the Worst Day in 22 years (see the CREME96',
     &       ' environment model)'
     &  /,3X,'included a stormy magnetosphere.',
     &      ' For many orbits, the generic stormy GTF'
     &  /,3x,'calculated here',
     &       ' can be substantially smaller than the actual GTF.')

 420      FORMAT(/,1X,'Enter altitude at apogee (kilometers):  ')
 400      FORMAT(/,1X,'Enter altitude at perigee (kilometers):  ')
 430      FORMAT(/,1X,'Input apogee < perigee, have been interchanged.')

 405      FORMAT(/,1X,'Enter orbital inclination (degrees):  ')

 409      FORMAT(/,1X,'The remaining input parameters are most relevant',
     &    ' to situations in which the',
     &  /,1X,'actual orbital path is known',
     &       ' or in which mission critical operations are',
     &  /,1x,'planned.',
     & //,3X,'[Recommended values are 0.0, unless you wish to examine',
     &  /,3X,'a very specific orbital segment.]')

 410      FORMAT(/,1X,'Enter initial longitude of ascending node',
     &            1X,'[Recommended = 0.0 (degrees)]:')
 415      FORMAT(/,1X,'Enter initial displacement from ascending',
```

```
     1          ' node',1X,'[Recommended = 0.0 (degrees)]:')
425   FORMAT(/,1X,'Enter displacement of perigee from',
     1  ' ascending node',1X,'[Recommended = 0.0 (degrees)]:')

426   FORMAT(/,1X,'Note:  for studies sensitive to a specific',
     &  ' orbital segment, you should be',
     &  /,1X,'aware that the GTF',
     &  ' calculations are averaged over 7 days at present.  This',
     &  /,1X,'parameter can be easily reset by modifying',
     &  ' the GEOMAG96 subroutine, but is ',
     &  /,1x,'not provided as a general-use input parameter.')


427   FORMAT(/,1X,'Enter name of output GTF file:',
     &          /,'    [Recommended:  something.GTF]')
428   FORMAT(A80)

450   FORMAT(/,1X,'Enter the number of desired GTF L-value bins ',
     &       '(1 - 10):',
     &  /,3X,'[Recommended default = 0, i.e. ',
     &       'one GTF for the entire orbit.]')

451   FORMAT(/,1X,
     &       'Enter the lower limits of the ',I2,' L-value bins: ',
     &  /,3X,'[A typical scenario could be to request 4 bins as the',
     &       ' as the previous entry.',
     &  /,3X,'Then, entries of 0.0, 2.0, 4.0, and 6.0',
     &       ' would subdivide the orbit into',
     &  /,3x,'sections with L < 2, L = 2-4, L = 4-6, and L > 6.]',
     & //,1X,'NOTE:  The L-value is a magnetic coordinate roughly',
     &       ' corresponding to the',
     &  /,1X,'distance in Earth Radii to the',
     &       ' magnetic field line at the magnetic equator.',
     &  /,1x,'For example, a geosynchronous orbit is roughly L = 6.6,',
     &       ' the geographic equator',
     &  /,1x,'is about L = 1, and the heart of the',
     &     ' South Atlantic Anomaly (SAA) is roughly at',
     &  /,1x,'L = 1.2 - 2.',
     &  '  Calculated L-values slightly less than 1 do occur; using',
     &  /,1X,'a lower limit of L = 0 will account for these.')

452   FORMAT(1X,'The L-values MUST be entered in increasing order',
     &  /,1X,'the L-value of ',F10.2,' has been reset to ',F10.2)

453   FORMAT(1X,'The number of L-values bins has been reset',
     &  /,1X,' from ',I2,' to ',I2)

454   FORMAT(/,1X,
     &       'Enter the decimal year for the field model in the ',
     &       'L-value calculations:',
     &  /,1X,' [Since the present IGRF grid calculations were performed',
     &       ' for 1980.0, that date',
     &  /,1X,' is presently recommended for consistency.]')

455   FORMAT(/,1X,'Enter root name of output GTF files:',
     &  /,1X,'[NOTE:  There will be ',I2,' output files, and',
     &     ' the files for the different L-value'
     &  /,1x,' bins will',
     &       ' be called something.GT# (# = 1,2,...,9,X)]')
```

```
456    FORMAT(1X,'Only 10 L-values are allowed.')

458    FORMAT(1X,'Calculation reset to whole orbit option, since',
    &          1X,'choosing 1 L bin',
    &        /,1X,' with a minimum L-value equal to 0 is equivalent to',
    &          1X,'the entire orbit.')

       END   !GTFDriverInput routine
```

```
      SUBROUTINE HEAVY_ION_UPSETS(LET_FILE,XM,YM,ZM,FUNNELM,
     &                            IPARAM,PARAMS,
     &                            XSECT_FILE,NBITS,IENTER,
     &                            SEU_RATE,DAY_RATE,PERSECOND,PERDAY)

C
C     Subroutine for performing heavy-ion evaluation:

C     Inputs:  LET_FILE = file containing integral LET spectrum
C                           (in ions/m2-s-sr) vs. LET (in MeV-cm2/g)
C              XM,YM,ZM = bit dimensions (in microns)
C              FUNNELM  = funnel length (optional; default 0) in microns
C              IPARAM      = 1,2,4, indicating cross-section model
C                             1 = Bendel 1-parameter
C                             2 = Bendel 2-parameter
C                             4 = Weibull
C                             5 = Critical charge (pc)
C                             0 = table
C              PARAMS(4)       = array containing cross-section parameters
C              XSECT_FILE = file containing cross-section table.
C              NBITS           = no. bits in the device:
C
C     Outputs:  SEU_RATE    in SEUs/bit/second
C               DAY_RATE    in SEUs/bit/day
C               PERSECOND   in SEUs/device/second
C               PERDAY      in SEUs/device/day
C
C     Written by:     Allan J. Tylka
C                     Code 7654
C                     Naval Research Laboratory
C                     Washington, DC 20375-5352
C                     tylka@crs2.nrl.navy.mil
C
C
C------------------------------------------------------------------------
C
      IMPLICIT NONE
      INTEGER*4 NBINS,NPTS,IPARAM,K,IENTER,MPTS
      REAL*4 LET,LETMG,FLUX,PARAMS,XM,YM,ZM,FUNNELM,XSECT,QC
      REAL*4 NBITS,SEU_RATE
      REAL*4 DAY_RATE,PERSECOND,PERDAY
      CHARACTER*80 LET_FILE,XSECT_FILE

      PARAMETER (NBINS=5000)
      DIMENSION LET(NBINS),LETMG(NBINS),FLUX(NBINS),XSECT(NBINS)
      DIMENSION PARAMS(4)
C

      WRITE(6,9998)
 9998 FORMAT(1x,' HI_UPSET_DRIVER calculation started.',
     &          '  Please stand by.')

      SEU_RATE=0.0

C     On first entry, get integral LET spectrum:

      IF (IENTER.EQ.1) THEN

      CALL UNLOAD_LET_SPECTRUM(LET_FILE,LET,FLUX,NPTS)
```

```
C        LET in spectrum file is in MeV-cm2/g; but cross-section evaluation
C        expects it in MeV-cm2/mg.

         DO 100 K=1,NPTS
            LETMG(K)=LET(K)*0.001
 100     CONTINUE

         ENDIF


         IF (XM*YM.GT.0.0) THEN

         IF (IPARAM.NE.5) THEN
C
C            For devices in which the cross-section has not reached its
C            limiting value at effective LET = LET(NPTS), the cross-section
C            table must be extended to higher effective LETs to MPTS:

             CALL EXTEND_EFFECTIVE_LET_RANGE(NPTS,NBINS,
     &                                       XM,YM,ZM,FUNNELM,MPTS,
     &                                       LET,LETMG,FLUX)


C            Evaluate SEU cross-section at these effective LET values:

             CALL EVALUATE_SEU_CROSS_SECTION(LETMG,MPTS,IPARAM,PARAMS,
     &                                       XSECT_FILE,XSECT)

C            Calculate SEU rate:

             CALL INTEGRATE_HEAVY_ION_UPSETS(MPTS,LET,FLUX,XSECT,
     &                                       XM,YM,ZM,FUNNELM,
     &                                       SEU_RATE)



         ELSEIF (IPARAM.EQ.5) THEN

             QC=PARAMS(1)
             CALL GET_UPSET(XM,YM,ZM,FUNNELM,QC,NPTS,LET,FLUX,SEU_RATE)
C            Re-scale to allow for possibility of different limiting
C            cross-section, rather than the customarily used XM*YM
             SEU_RATE=SEU_RATE*PARAMS(2)/XM/YM

         ENDIF

         ENDIF

         IF (SEU_RATE.LT.0.) THEN
             WRITE(6,999) SEU_RATE
 999         FORMAT(1x,' ERROR in HEAVY_ION_UPSETS: SEU RATE = ',E13.5)
             SEU_RATE=0.0
         ENDIF


         CALL CALC_SEU_RATE(NBITS,SEU_RATE,DAY_RATE,PERSECOND,PERDAY)

         WRITE(6,9999)
 9999    FORMAT(1x,' HI_UPSET_DRIVER calculation completed. ')
```

```
      RETURN
      END


      SUBROUTINE EXTEND_EFFECTIVE_LET_RANGE(NPTS,NBINS,
     &                                     XM,YM,ZM,FUNNELM,MPTS,
     &                                     LET,LETMG,FLUX)

C     Based on device dimesions, extends the range of effective LET
C     values from NPTS to MPTS
C
      IMPLICIT NONE
      INTEGER*4 NPTS,NBINS,MPTS
      REAL*4 XM,YM,ZM,FUNNELM
      REAL*4 LET,LETMG,FLUX
      DIMENSION LET(1),LETMG(1),FLUX(1)
      INTEGER*4 K,NLAST,NEXTRA
      REAL*4 FACTOR,DL

      MPTS=NPTS

C     Locate last non-zero integral flux point:
      NLAST=0
      DO 100 K=1,NPTS
         IF (FLUX(K).LE.0.0 .and. NLAST.EQ.0) NLAST=K
 100  CONTINUE

      FACTOR=(SQRT(XM*XM+YM*YM+ZM*ZM)+FUNNELM)/ZM
      DL=ALOG(LET(NPTS)/LET(1))**(1./FLOAT(NPTS-1))
      NEXTRA=1.+ALOG(FACTOR)/ALOG(DL)
      MPTS=NEXTRA+NLAST
      IF (MPTS.GT.NBINS) MPTS=NBINS
      IF (MPTS.GT.NPTS) THEN
      DO 200 K=NPTS+1,MPTS
             LET(K)=LET(NPTS)*DL**(K-NPTS)
             LETMG(K)=LET(K)*0.001
             FLUX(K)=0.0
 200  CONTINUE
      ENDIF
C
C     Debug:
C      TYPE *,' NPTS,   LET(NPTS):   ',NPTS,LET(NPTS)
C      TYPE *,' NLAST,  LET(NLAST): ',NLAST,LET(NLAST)
C      TYPE *,' FACTOR,NEXTRA,MPTS: ',FACTOR,NEXTRA,MPTS
C      TYPE *,' LET(MPTS):          ',LET(MPTS)

      RETURN
      END
```

```
      PROGRAM HI_UPSET_DRIVER

      IMPLICIT NONE
      REAL*4 XM,YM,ZM,FUNNELM,NBITS,PARAMS
      REAL*4 XM0,YM0,ZM0
      REAL*4 SEU_RATE,DAY_RATE,PERSECOND,PERDAY
      INTEGER*4 IPARAM,IREPEAT,IENTER
      DIMENSION PARAMS(4)
      CHARACTER*80 LET_FILE,XSECT_FILE,REPORT_FILE
      CHARACTER*40 DEVICE_LABEL
      INTEGER*4 IERR
      DATA IERR/0/
      INTEGER*4 IENT
      DATA IENT/0/
C
C     Modified 11/8/96: to extract XM,YM from cross-section data
C                       if user supplies XM=YM=0.
C
      IENTER=1

 10   CONTINUE
      CALL INITIALIZE_HI_UPSETS(LET_FILE,XM0,YM0,ZM0,FUNNELM,NBITS,
     &                          IPARAM,PARAMS,XSECT_FILE,IENTER,
     &                          DEVICE_LABEL,REPORT_FILE)

      CALL CHECK_RPP_DIMENSIONS(XM0,YM0,ZM0,
     &                          IPARAM,PARAMS,XSECT_FILE,
     &                          XM,YM,ZM)


      CALL HEAVY_ION_UPSETS(LET_FILE,XM,YM,ZM,FUNNELM,IPARAM,PARAMS,
     &                      XSECT_FILE,NBITS,IENTER,
     &                      SEU_RATE,DAY_RATE,PERSECOND,PERDAY)

      CALL HI_UPSET_REPORT(LET_FILE,XM,YM,ZM,FUNNELM,NBITS,
     &                     IPARAM,PARAMS,XSECT_FILE,IENTER,
     &                     DEVICE_LABEL,REPORT_FILE,
     &                     SEU_RATE,DAY_RATE,PERSECOND,PERDAY)


 9100     CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,9200)
 9200     FORMAT(//,' Repeat SEU rate calculation with different',
     &            ' device characteristics? (1=yes,0=no)')
          READ(*,*,ERR=9100,IOSTAT=IERR) IREPEAT
          IF (IREPEAT.EQ.1) THEN
              IENTER=IENTER+1
              GOTO 10
          ENDIF

      WRITE(6,9600)
 9600 FORMAT(1x,' Heavy Ion Upset Calculations finished.')
      STOP
      END
```

```
      SUBROUTINE HI_UPSET_REPORT(LET_FILE,XM,YM,ZM,FUNNEL,NBITS,
     &                           IPARAM,PARAMS,XSECT_FILE,IENTER,
     &                           DEVICE_LABEL,REPORT_FILE,
     &                           SEU_RATE,DAY_RATE,PERSECOND,PERDAY)


      IMPLICIT NONE
      REAL*4 XM,YM,ZM,FUNNEL,NBITS,PARAMS
      REAL*4 SEU_RATE,DAY_RATE,PERSECOND,PERDAY
      INTEGER*4 IPARAM,IENTER,OUTUNIT,VERSION_NUMBER,NHEADER0,K
      INTEGER*4 NHEADER,PROGRAM_CODE,STAT,CREME96_OPEN
      DATA OUTUNIT/2/
      DIMENSION PARAMS(4)
      CHARACTER*80 LET_FILE,XSECT_FILE,REPORT_FILE
      CHARACTER*40 DEVICE_LABEL
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME

      PROGRAM_CODE=10

      IF (IENTER.EQ.1.and.REPORT_FILE.NE.'NULLFILE') THEN
c         OPEN(UNIT=OUTUNIT,FILE='USER:'//REPORT_FILE,STATUS='NEW')
       stat = creme96_open(report_file,'user',outunit,'new')
       CALL DATE(CREATION_DATE)
       CALL TIME(CREATION_TIME)
       CALL GET_CREME96_VERSION(VERSION_NUMBER)
       CALL CHECK_HEADER_LENGTH(LET_FILE,NHEADER0)
       NHEADER=NHEADER0+2
       WRITE(OUTUNIT,991) NHEADER,REPORT_FILE(1:70),
     &                    VERSION_NUMBER,PROGRAM_CODE
 991   FORMAT(I3,1x,A70,I4,I2)
       WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME
 992   FORMAT(1x,'%Created by CREME96:HI_UPSET_DRIVER Version ',I4,
     &         ' on ',A9,' at ',A8)

C      Now copy header information from input file:
       WRITE(OUTUNIT,993) LET_FILE(1:40)
 993   FORMAT(1x,'%Input Integral LET Spectrum File: ',A40)
       CALL COPY_HEADERS(LET_FILE,NHEADER0,OUTUNIT)
      ENDIF

      IF (REPORT_FILE.NE.'NULLFILE') THEN
       WRITE(OUTUNIT,994) IENTER,DEVICE_LABEL
 994   FORMAT(/,1x,' REPORT NO. ',I4,': ',2x,A40)
       WRITE(outunit,995) XM,YM,ZM,FUNNEL
 995   FORMAT(1x,' RPP Dimensions: X = ',F10.5,'  Y = ',F10.5,
     &             '  Z = ',F10.5,' microns.',
     &       /,1x,' Funnel length = ',F10.5,' microns.')
       IF (IPARAM.EQ.0) WRITE(outunit,980) IPARAM,XSECT_FILE(1:75)
       IF (IPARAM.EQ.1) WRITE(outunit,981) IPARAM,PARAMS(1)
       IF (IPARAM.EQ.2) WRITE(outunit,982) IPARAM,PARAMS(1),PARAMS(2)
       IF (IPARAM.EQ.4) WRITE(outunit,984) IPARAM,(PARAMS(K),K=1,4)
       IF (IPARAM.EQ.5) WRITE(outunit,985) IPARAM,PARAMS(1),PARAMS(2)
       WRITE(outunit,996) NBITS
 996   FORMAT(1x,' Number of bits = ',E13.5)

 980   FORMAT(1x,' CROSS-SECTION INPUT ',I3,'  FROM FILE: ',
     &       /,5x,A75)
 981   FORMAT(1x,' CROSS-SECTION INPUT ',I3,
     &             '  BENDEL 1-PARAMETER = ',E13.5)
```

```
982    FORMAT(1x,' CROSS-SECTION INPUT ',I3,
   &             '  BENDEL 2-PARAMETERS A,B = ',2E13.5)
984    FORMAT(1x,' CROSS-SECTION INPUT ',I3,
   &             '  WEIBULL FIT: ',
   &        /,5x,' ONSET   = ',F9.3,' MeV-cm2/milligram',
   &        /,5x,' WIDTH   = ',F9.3,' MeV-cm2/milligram',
   &        /,5x,' POWER   = ',F9.3,' (dimensionless)',
   &        /,5x,' PLATEAU = ',F9.3,' square microns/bit')
985    FORMAT(1x,' CROSS-SECTION INPUT ',I3,
   &        /,5x,' Critical charge = ',E13.5,' picocoloumbs',
   &        /,5x,' Cross-Section   = ',E13.5,' square microns/bit')


       WRITE(outunit,9200)
       WRITE(outunit,9201) IENTER,SEU_RATE,DAY_RATE,PERSECOND,PERDAY
9200   FORMAT(2x,'Rates:       SEUs/bit/second      /bit/day',
   &              '     /device/second     /device/day')
9201   FORMAT(2x,'*****',I4,2x,4(E14.5,2x))
       ENDIF

       WRITE(6,9200)
       WRITE(6,9201) IENTER,SEU_RATE,DAY_RATE,PERSECOND,PERDAY

      RETURN
      END
```

```
      SUBROUTINE indexx(n,NMAX,arr,indx)
      IMPLICIT NONE
C     Explicit variable lengths added AJT 12-12-96
C      INTEGER n,indx(n),M,NSTACK
C      REAL arr(n)
C      PARAMETER (M=7,NSTACK=50)
C      INTEGER i,indxt,ir,itemp,j,jstack,k,l,istack(NSTACK)
C      REAL a
C     Parameter NMAX added to define size of passed-in arrays.
      INTEGER*4 n,NMAX,indx(1),M,NSTACK
      REAL*4 arr(1)
      PARAMETER (M=7,NSTACK=50)
      INTEGER*4 i,indxt,ir,itemp,j,jstack,k,l,istack(NSTACK)
      REAL*4 a
      IF (N.GT.NMAX) THEN
          WRITE(6,9999) N,NMAX
 9999     FORMAT('@ 99999 ABNORMAL TERMINATION: ',
     &      /,1x,' Error in INDEXX: N,NMAX: ',2I12,
     &      /,1x,' STOP.')
          STOP
      endif
      do 11 j=1,n
        indx(j)=j
11    continue
      jstack=0
      l=1
      ir=n
1     if(ir-l.lt.M)then
        do 13 j=l+1,ir
          indxt=indx(j)
          a=arr(indxt)
          do 12 i=j-1,1,-1
            if(arr(indx(i)).le.a)goto 2
            indx(i+1)=indx(i)
12        continue
          i=0
2         indx(i+1)=indxt
13      continue
        if(jstack.eq.0)return
        ir=istack(jstack)
        l=istack(jstack-1)
        jstack=jstack-2
      else
        k=(l+ir)/2
        itemp=indx(k)
        indx(k)=indx(l+1)
        indx(l+1)=itemp
        if(arr(indx(l+1)).gt.arr(indx(ir)))then
          itemp=indx(l+1)
          indx(l+1)=indx(ir)
          indx(ir)=itemp
        endif
        if(arr(indx(l)).gt.arr(indx(ir)))then
          itemp=indx(l)
          indx(l)=indx(ir)
          indx(ir)=itemp
        endif
        if(arr(indx(l+1)).gt.arr(indx(l)))then
          itemp=indx(l+1)
          indx(l+1)=indx(l)
```

```
            indx(l)=itemp
          endif
          i=l+1
          j=ir
          indxt=indx(l)
          a=arr(indxt)
3         continue
            i=i+1
          if(arr(indx(i)).lt.a)goto 3
4         continue
            j=j-1
          if(arr(indx(j)).gt.a)goto 4
          if(j.lt.i)goto 5
          itemp=indx(i)
          indx(i)=indx(j)
          indx(j)=itemp
          goto 3
5         indx(l)=indx(j)
          indx(j)=indxt
          jstack=jstack+2
          if(jstack.gt.NSTACK)pause 'NSTACK too small in indexx'
          if(ir-i+1.ge.j-l)then
            istack(jstack)=ir
            istack(jstack-1)=i
            ir=j-1
          else
            istack(jstack)=j-1
            istack(jstack-1)=l
            l=i
          endif
        endif
        goto 1
        END
```

```
      SUBROUTINE INIFLUX(IZMIN,IZMAX,EMIN,EMAX,YEAR,IMODE,ITRANS,
     *                   GTRANSFILE,TRAPDFILE,FLXFILE)

C
C     Subroutine for initializing input parameters to CREME96 environment
C     model.
C
C     Modified 9/12/96: Energy range fixed at 0.1-1.0E+5 MeV/nuc;
C                       However, the external flux routines return 0 for
C                       E < 1.0 MeV/nuc; the 0.1 threshold is put in here
C                       for subsequent tracking through shielding.
C
C
C     Modified 11/18/97: Allow input of trapped proton file.
C
C
      IMPLICIT NONE
      INTEGER*4 IMINTEMP,IMAXTEMP,IACCEPT,IFILETYPE
      INTEGER*4 IZMIN,IZMAX,IMODE,ITRANS,ITYPE,ITRP
      INTEGER*4 ISEPMODE
      REAL*4 EMIN,EMAX,YEAR,YEARDUM
      REAL*4 EMINTEMP,EMAXTEMP
      CHARACTER*80 GTRANSFILE,TRAPDFILE,FLXFILE
      CHARACTER*1 IBLANK
      DATA IBLANK/' '/
      INTEGER*4 IERR
      DATA IERR/0/
C

      WRITE(6,1000)
 1000 FORMAT(' CREME96 IONIZING RADIATION ENVIRONMENT MODEL',/,
     &        ' ----> FLUX_DRIVER Module: External Environment')
      WRITE(6,1001)
 1001 FORMAT(
     &  ' This program will calculate the particle environment',
     &  ' outside of the spacecraft.',/,
     &  ' You must run additional programs after this to',
     &  ' (1) transport the particles'
     & /,' through shielding; and (2) calculate SEU rates.',
     &//,' BEFORE RUNNING THIS OR ANY OTHER CREME96 PROGRAM',
     &  ' PLEASE DEFINE THREE LOGICALS:',/,
     & /,4x,' CREME96       as the directory where CREME96 source',
     &        ' & executables reside.',
     & /,4x,' CR96TABLES    as the directory in which CREME96 data',
     &        ' tables reside.',
     & /,4x,' USER          as the directory in which output files',
     &        ' should be written.',
     & //,' Now begin specification of the environment parameters: ')

  101 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1002)
 1002 FORMAT(/,' Enter minimum & maximum atomic numbers: ',
     &   /,' Recommended for most applications: ',
     &     ' IZMIN = 1 (hydrogen) to IZMAX = 28 (nickel).',
     &   /,'    [Enter 0 0 <CARRIAGE RETURN> for recommended values.] ',
     &   /,'    NOTE: For >95% of all SEU applications, Z > 28',
     &     ' elements, which are very',
     &   /,'    rare, may be neglected.  However, for SEU rates in',
     &     ' devices with high',
```

```
     &   /,'    thresholds (> 15 MeV-cm2/mg) these heavier elements',
     &     ' MAY be important,',
     &   /,'    particularly for low-inclination low-Earth orbits',
     &     ' or for applications',
     &   /,'    demanding very low SEU rates.  Please note that',
     &     ' including Z > 28 elements in',
     &   /,'    you calculations can',
     &     ' significantly slow down some parts of the CREME96 code.')
       READ(*,*,ERR=101,IOSTAT=IERR) IMINTEMP,IMAXTEMP
       IZMAX=MAX(IMINTEMP,IMAXTEMP)
       IZMIN=MIN(IMINTEMP,IMAXTEMP)
       IF (IZMIN.EQ.0 .and. IZMAX.EQ.0) THEN
           IZMIN=1
           IZMAX=28
       ENDIF


       IF (IZMIN.LE.0 .or. IZMAX.GT.92) THEN
           WRITE(6,9001) IZMIN,IZMAX
 9001      FORMAT(1x,' Invalid atomic number(s): ',2I5,
     &          /,1x,' Please try again.')
           GOTO 101
       ENDIF

       WRITE(6,9002) IZMIN,IZMAX
 9002  FORMAT(1x,'  Lowest  atomic number = ',I5,
     &      /,1x,'  Highest atomic number = ',I5)

  103  CONTINUE
       EMIN=0.1
       EMAX=1.0E+5
C-----------------------------------------------------------------------------
C      Following code for specifying energy interval obsolete 9/12/96:
C       CALL RETRY_INPUT(IERR)
C       WRITE(6,1003)
C 1003  FORMAT(/,' Enter minimum & maximum energy (in MeV/nuc): ',
C     &        /,' Recommended for most SEE applications: ',
C     &        '  EMIN = 10.0; EMAX = 1.0E+5')
C       TYPE *,' [Enter 0 0 <CARRIAGE RETURN> for recommended values.]'
C       READ(*,*,ERR=103,IOSTAT=IERR) EMINTEMP, EMAXTEMP
C       EMIN=MIN(EMINTEMP,EMAXTEMP)
C       EMAX=MAX(EMINTEMP,EMAXTEMP)
C       IF (EMAX.LE.0.0 .and. EMIN.LE.0.0) THEN
C       ENDIF
C       IF (EMIN.LE.0 .or. EMAX.LE.0.0 .or. EMIN.EQ.EMAX) THEN
C           TYPE *,' Invalid energy value(s): ',EMIN,EMAX
C           TYPE *,' Please try again.'
C           GOTO 103
C       ENDIF
C       TYPE *,'  Minimum energy =  ',EMIN,' MeV/nuc'
C       TYPE *,'  Maximum energy =  ',EMAX,' MeV/nuc'
C-----------------------------------------------------------------------------
  104  CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,1004)
 1004  FORMAT(/,' Specify type of environment model: ',
     &         ' Enter 0 or 1: ',
     &    /,'   0 = Solar-quiet (ie., no Solar Energetic Particles)',
     &    /,'   1 = Solar Energetic Particles ONLY')
```

```
      WRITE(6,1041)
1041  FORMAT('    NOTE: Choosing 1 (Solar Energetic Particles ONLY)',
     &       ' does not include',
     &     /,'    Galactic cosmic rays, which may also contribute',
     &       ' to the SEU rate behind',
     &     /,'    thick shielding during a solar particle event.')

      READ(*,*,ERR=104,IOSTAT=IERR) ITYPE

      IF (ITYPE.NE.0 .and. ITYPE.NE.1) THEN
          WRITE(6,9010) ITYPE
9010      FORMAT(1x,' Environment type ',I6,' unknown.',
     &          /,1x,' Please try again.')
          GOTO 104
      ENDIF

      GTRANSFILE='           '
      TRAPDFILE ='           '

105   CONTINUE
      CALL RETRY_INPUT(IERR)
      IF (ITYPE.EQ.0) THEN
      IMODE=0
      WRITE(6,1005)
1005  FORMAT(/,' Solar-quiet period.  Enter decimal year',
     &         ' (eg. 1996.42) OR ',
     &  /,3x,'0 for Solar Minimum (Cosmic-Ray Maximum, YEAR =1977.0)',
     &  /,3x,'1 For Solar Maximum (Cosmic-Ray Minimum, YEAR =1990.2)')

      READ(*,*,ERR=105,IOSTAT=IERR) YEARDUM

      IF (ABS(YEARDUM)     .LE. 0.0001) THEN
       YEAR=1977.0
       WRITE(6,9020) YEAR
9020   FORMAT(1x,' Solar Minimum (Cosmic-Ray Maximum) YEAR = ',F10.3)
      ELSEIF (ABS(YEARDUM-1.0) .LE. 0.0001) THEN
       YEAR=1990.2
       WRITE(6,9021) YEAR
9021   FORMAT(1x,' Solar Maximum (Cosmic-Ray Minimum) YEAR = ',F10.3)
      ELSE
       YEAR=YEARDUM
       WRITE(6,9022) YEAR
9022   FORMAT(1x,' YEAR  = ',F10.3)
      ENDIF

      ELSEIF (ITYPE.EQ.1) THEN

      YEAR=0.0

 106  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1006)
1006  FORMAT(/,' CREME96 currently provides three Solar Energetic',
     &       ' Particle Models: ',
     &  /,3x,' Worst Week in 22 years:  based on observed proton and',
     &       ' heavy-ion fluences',
     &  /,28x,' on 19-26 October 1989;',
     &  /,3x,' Worst Day in 22 years:   based on observed proton and',
     &       ' heavy-ion fluences',
     &  /,28x,' on 20 October 1989',
```

```
     &   /,3x,' Peak Instantaneous Flux: based on peak 5-minute-'
     &        'average fluxes observed',
     &   /,28x,' during 19-26 October 1989.',
     &   /,3x,' Enter 1 for worst week; 2 for worst day;',
     &        ' 3 for peak flux: ')
      READ(*,*,ERR=106,IOSTAT=IERR) ISEPMODE
C
C     Sloppy coding, introduced here on 9/14/96: ISEPMODE gives
C     natural progression toward increasing severity, which is
C     incompatible with original definitions of IMODE.  Unfortunately,
C     the IMODE values are deeply imbedded in the code and I have
C     chosen not to change them at this time.
C
      IF (ISEPMODE.EQ.2) THEN
          IMODE=1
          WRITE(6,1007)
 1007     FORMAT(' Worst Day Solar Energetic Particle Model chosen.')
      ELSEIF (ISEPMODE.EQ.1) THEN
          IMODE=2
          WRITE(6,1008)
 1008     FORMAT(' Worst Week Solar Energetic Particle Model chosen.')
      ELSEIF (ISEPMODE.EQ.3) THEN
          IMODE=3
          WRITE(6,1081)
 1081     FORMAT(' Peak Solar Energetic Particle Flux Model chosen.')
      ELSE
          WRITE(6,1009)
 1009     FORMAT(' Requested SEP environment not defined.',
     &            ' Please try again.')
          GOTO 106
      ENDIF

      ENDIF

  107 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1010)
 1010 FORMAT(/,' Specify Environment Location: ',
     &          ' Enter 0 or 1: ',
     &        /,'  0 = Interplanetary Space near Earth',
     &        /,'       (ie., outside of Earths magnetosphere)'
     &        /,'  1 = Inside Earths magnetosphere',
     & /,'       (You will need to supply a geomagnetic transmission',
     &         ' function file.',
     & /,'        Run GTRANS_DRIVER to make one.)')
      READ(*,*,ERR=107,IOSTAT=IERR) ITRANS

      IF (ITRANS.NE.0 .and. ITRANS.NE.1) THEN
          WRITE(6,9030) ITRANS
 9030     FORMAT(1x,' Environment location ',I5,' unknown.',
     &            '  Please try again.')
          GOTO 107
      ENDIF

      IF (ITRANS.EQ.0) THEN
          WRITE(6,1011)
 1011     FORMAT(' Geosynchronous Orbit or'
     &            ' Near-Earth Interplanetary Space')
      ELSEIF (ITRANS.EQ.1) THEN
  112     CONTINUE
```

```
          CALL RETRY_INPUT(IERR)
          WRITE(6,1012)
 1012     FORMAT(' Inside Earths Magnetosphere')
          WRITE(6,1013)
 1013     FORMAT(' Specify name of geomagnetic transmission file: ',
     &         /,' ie., something.GTF')
          READ(*,1014,ERR=112,IOSTAT=IERR) GTRANSFILE
 1014     FORMAT(A80)

          IF (GTRANSFILE.EQ.IBLANK) THEN
          WRITE(6,1914)
 1914      FORMAT(1x,' You must specify a geomagnetic transmission',
     &                 ' file (.GTF) for the calculation you',
     &          /,1x,' have outlined. Please try again.',/)
           GOTO 112
          ELSE
           IFILETYPE=2
           WRITE(6,1015) GTRANSFILE
 1015      FORMAT(' Geomagnetic Transmission File =',/,1x,A80)
           CALL CHECK_FILE(IFILETYPE,GTRANSFILE,IACCEPT)
           IF (IACCEPT.NE.0) GOTO 112
          ENDIF

          ITRP=0
          IF (IZMIN.EQ.1) THEN
          IF (IMODE.EQ.0) THEN
  116     CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,1016)
 1016     FORMAT(' Include Trapped Protons? (0=no; 1=yes)')
C-----------------------------------------------------------------------------
C          WRITE(6,9999)
C 9999     FORMAT(1x,' *** NOTE:  This test version of CREME96',
C    &                 ' does NOT include trapped protons.',
C    &          /,1x,' Please enter 0')
C-----------------------------------------------------------------------------
          READ(*,*,ERR=116,IOSTAT=IERR) ITRP
          ENDIF

          IF (ITRP.EQ.0) THEN
              WRITE(6,1027)
 1027         FORMAT(1x,' No Trapped Protons Included.')
          ELSEIF (ITRP.EQ.1) THEN
  117         CONTINUE
              CALL RETRY_INPUT(IERR)
              WRITE(6,1017)
 1017         FORMAT(' Trapped Protons included.',
     &          /,1x,' Enter name of',
     &        ' file containing orbit-averaged trapped proton flux:')
              READ(*,1014,ERR=117,IOSTAT=IERR) TRAPDFILE


              IF (TRAPDFILE.EQ.IBLANK) THEN
                 WRITE(6,1917)
 1917            FORMAT(1x,' You must specify a trapped proton',
     &             ' file for the calculation you have outlined.',
     &             /,1x,' Please try again.',/)
                 GOTO 117
              ELSE
```

```
            IFILETYPE=1
            WRITE(6,1018) TRAPDFILE
1018        FORMAT(' Trapped Proton Flux File =',/,1x,A80)
            CALL CHECK_FILE(IFILETYPE,TRAPDFILE,IACCEPT)
            IF (IACCEPT.NE.0) GOTO 117
            ITRANS=2
         ENDIF

      ENDIF
      ENDIF
      ENDIF

 119  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1019)
1019  FORMAT(/,' Particle environment specification now completed.',
     &       /,' Enter name of output file: '
     &       /,' Note: for standard CREME96 format, must be',
     &         ' something.FLX')

120   CONTINUE
      READ(*,1014,ERR=119,IOSTAT=IERR) FLXFILE
      WRITE(6,1020) FLXFILE
1020     FORMAT(' Output FLux File =',/,1x,A80)

      CALL CHECK_OUTPUT_FILE(FLXFILE,IACCEPT)
      IF (IACCEPT.NE.0) GOTO 120

      RETURN
      END
```

```
      SUBROUTINE INIDOSE(INFILE,LETMINMG,LETMAXMG,
     &                      IZMIN,IZMAX,EMINCUT,EMAXCUT,MATERIAL,OUTFILE)
C
C     Subroutine for initializing input parameters to the DOSE program
C     in CREME96.  This version only allows SILICON devices.
C
      IMPLICIT NONE
      INTEGER*4 IZMIN,IZMAX,IMINTEMP,IMAXTEMP,IFILETYPE,IACCEPT
      REAL*4 LETMINMG,LETMAXMG,LETMINTEMP,LETMAXTEMP
      REAL*4 EMINCUT,EMAXCUT
      CHARACTER*80 INFILE,OUTFILE,DEFAULT_NAME
      CHARACTER*12 MATERIAL
      CHARACTER*1 IBLANK
      DATA IBLANK/' '/
      INTEGER*4 IERR,IDIFSPEC,ILONG
      DATA IERR/0/
C
      WRITE(6,1000)
 1000 FORMAT(' CREME96 IONIZING RADIATION ENVIRONMENT MODEL',
     & /,' -->  Ionizing Dose Calculation')
      WRITE(6,1001)
 1001 FORMAT(' This program will calculate the dose resulting from',
     &   ' CREME96 differential',
     & /,' particle fluxes.  This program is intended primarily for',
     &   ' calculating dose due to',
     & /,' NON-TRAPPED components of the radiation environment',
     &   ' [cosmic rays and solar',/,' energetic',
     &   ' (flare) particles].',
     &   '  This program is NOT recommended for calculating',
     & /,' dose due to TRAPPED particles,',
     &  ' which generally dominate the dose inside',
     & /,' Earths magnetosphere. ',
     &   ' CREME96 does NOT included trapped electrons, and',
     & /,' trapped-proton dose is more accurately described',
     &   ' by other programs, especially',
     & /,' for lightly-shielded systems.',
     &//,' Before running this program, you must do:',
     &//,'    FLUX',
     &   '  to generate the particle environment outside',
     &   ' the spacecraft; and',
     & /,'    TRANS',
     &   ' to transport fluxes through the spacecraft shielding.',
     &//,' NOTE: Before running this or any other CREME96 program',
     &   ' please define 3 logicals:',/,
     & /,4x,' CREME96      as the directory where CREME96 source',
     &        ' & executables reside.',
     & /,4x,' CR96TABLES   as the directory in which CREME96 data',
     &        ' tables reside.',
     & /,4x,' USER         as the directory in which output files',
     &        ' should be written.',
     &  //,' Now begin specification of the DOSE_DRIVER inputs: ')

       INFILE='      '

  112  CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,1002)
 1002  FORMAT(' Enter name of file containing',
     &           ' CREME96 particle fluxes: '
     &        /,' ie. something.TFX from TRANS or',
```

```
     &         /,'      (for zero shielding) something.FLX from FLUX'
     &           ,' or something.TR* from TRP:')
      READ(*,1014,ERR=112,IOSTAT=IERR) INFILE
1014  FORMAT(A80)

      IF (INFILE.EQ.IBLANK) THEN
          WRITE(6,1914)
1914      FORMAT(1x,' You must specify an input file here,',
     &                ' either from TRANS or',
     &    /,1x,' (in the case of zero shielding) from',
     &         ' FLUX or TRP. ',
     &                ' Please try again.',/)
           GOTO 112
      ELSE
          IFILETYPE=4
          WRITE(6,1020) INFILE
1020      FORMAT(' Input Flux File =',/,1x,A80)
          CALL CHECK_FILE(IFILETYPE,INFILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 112
      ENDIF

 103  CONTINUE
C
C     Modification 9/12/96: LET range hardwired:
      LETMINMG=1.0E-3
      LETMAXMG=1.1E+2

1032  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1004)
1004  FORMAT(' Enter minimum & maximum atomic numbers to be',
     &          ' included in dose calculation:',
     &        /,'    [Enter 0 0 <CARRIAGE RETURN> for full range',
     &          ' included in the input flux file.]')

101   CONTINUE
      READ(*,*,ERR=1032,IOSTAT=IERR) IMINTEMP,IMAXTEMP

      IF (IMAXTEMP.NE.0 .and. IMAXTEMP.LT.IMINTEMP) THEN
          IZMIN=MIN(IMINTEMP,IMAXTEMP)
          IZMAX=MAX(IMINTEMP,IMAXTEMP)
      ELSE
          IZMIN=IMINTEMP
          IZMAX=IMAXTEMP
      ENDIF

      IF (IZMIN.LT.0 .or. IZMIN.GT.92
     &    .or. IZMAX.LT.0 .or. IZMAX.GT.92) THEN
              WRITE(6,9002) IZMIN,IZMAX
9002         FORMAT(1x,' Invalid atomic number(s): ',2I5,
     &              /,1x,' Please try again.')
              GOTO 101
      ENDIF

      IF (IZMIN.EQ.0 .and. IZMAX.EQ.0) THEN
          WRITE(6,1039)
1039  FORMAT(' Nominal Z range from input flux file used.')
      ELSEIF (IZMIN.EQ.0 .and. IZMAX.NE.0) THEN
          WRITE(6,1038) IZMAX
1038  FORMAT(' Minimum Z value from input flux file; Maximum Z =',I3)
```

```
      ELSEIF (IZMIN.NE.0 .and. IZMAX.EQ.0) THEN
          WRITE(6,1037) IZMIN
 1037 FORMAT(' Minimum Z =',I3,'; Maximum Z value from',
     &        ' input flux file.')
      ELSE
          WRITE(6,1040) IZMIN,IZMAX
 1040 FORMAT(1x,' Dose accumulated for elements',
     &      /,1x,I3,' </= Z </= ',I3)
      ENDIF

C     12/1/97: EMIN,EMAX hardwired.  Keep source code here in case requested
C     by beta-testers.
      EMINCUT=0.1
      EMAXCUT=1.0E+5
C-----------------------------------------------------------------------
C 105  CONTINUE
C      CALL RETRY_INPUT(IERR)
C      EMAXCUT=1.0E+24
C      WRITE(6,1005)
C 1005 FORMAT(' Enter minimum particle energy (in MeV/nuc)',
C    &        ' to be included in accumulating the'
C    &      /,' dose calculation:')
C
C       READ(*,*,ERR=105,IOSTAT=IERR) EMINCUT
C       IF (EMINCUT.LT.0.) THEN
C          WRITE(6,9005) EMINCUT
C 9005     FORMAT(1x,' Invalid minimum energy value: ',E13.6,
C    &          /,1x,' Please try again.')
C          GOTO 105
C       ENDIF
C
C      WRITE(6,1051) EMINCUT
C 1051 FORMAT(/,' Dose accumulated for',
C    &          ' nuclei with energy > ',F8.3,' MeV/nuc.')
C-----------------------------------------------------------------------

      MATERIAL='SILICON'
      WRITE(6,1007) MATERIAL
 1007 FORMAT(/,' Dose calculated in ',A12)

 1017 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1008)
 1008 FORMAT(/,' Enter name of output file: ',
     &      /,' Note: According to CREME96 naming conventions,',
     &        ' should be something.dse')

      ILONG=INDEX(INFILE,'.')
      IF (ILONG.NE.0) THEN
          DEFAULT_NAME=INFILE(1:ILONG)//'DSE'
      ELSE
          DEFAULT_NAME=INFILE//'.DSE'
      ENDIF

      WRITE(6,1028) DEFAULT_NAME(1:79)
 1028 FORMAT(' Suggested name:',/,1x,A79,
     &      /,' Hit RETURN if this is acceptable.')

 1018 CONTINUE
```

```
      READ(*,1014,ERR=1017,IOSTAT=IERR) OUTFILE
      IF (OUTFILE.EQ.IBLANK) OUTFILE=DEFAULT_NAME

      WRITE(6,1009) OUTFILE
1009  FORMAT(' Output Flux File =',/,1x,A80)

      CALL CHECK_NAME_CONFLICT(INFILE,OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) GOTO 1017

      CALL CHECK_OUTPUT_FILE(OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) THEN
      WRITE(6,1010) INFILE(1:75)
      WRITE(6,1011) OUTFILE(1:75)
      WRITE(6,1012)
1010  FORMAT(1x,' INPUT file = ',/,5x,A75)
1011  FORMAT(1x,' Previous try at OUTPUT name = ',/,5x,A75)
1012  FORMAT(1x,' Try again, ie., newname.DSE')
      GOTO 1018
      ENDIF

      RETURN
      END
```

```fortran
      SUBROUTINE INILET(INFILE,LETMINMG,LETMAXMG,
     &                    IZMIN,IZMAX,EMINCUT,EMAXCUT,MATERIAL,OUTFILE,
     &                    IDIFSPEC)

C
C     Subroutine for initializing input parameters to LETSPEC program
C     in CREME96.  This version only allows SILICON devices.
C
C     Modifications 11/8/96: Allow 0 for input IZ values to select
C                            either lowest or highest from input flux
C                            file.
C
C     Modification 10/31/96: option of differential LET spectrum added.
C
C     Modifications 9/12/96: LETMINMG,LETMAXMG hardwired;
C                            allow user to specify minimum energy in
C                            flux accumulation.
C
      IMPLICIT NONE
      INTEGER*4 IZMIN,IZMAX,IMINTEMP,IMAXTEMP,IFILETYPE,IACCEPT
      REAL*4 LETMINMG,LETMAXMG,LETMINTEMP,LETMAXTEMP
      REAL*4 EMINCUT,EMAXCUT
      CHARACTER*80 INFILE,OUTFILE,DEFAULT_NAME
      CHARACTER*12 MATERIAL
      CHARACTER*1 IBLANK
      DATA IBLANK/' '/
      INTEGER*4 IERR,IDIFSPEC,ILONG
      DATA IERR/0/
C
      WRITE(6,1000)
 1000 FORMAT(' CREME96 IONIZING RADIATION ENVIRONMENT MODEL',
     & /,' -->  INTEGRAL Linear Energy Transfer (LET)',
     & ' Spectrum Calculation')
      WRITE(6,1001)
 1001 FORMAT(' This program will transform the input differential',
     &   ' particle energy spectra from',
     & /,' CREME96 into an LET spectrum, ie.,',
     &   ' particle flux vs. LET [in MeV-cm2/g]',
     & /,' as appropriate for SEU calculations with CREME96.  Before',
     &   ' running this',
     & /,' program, you must do:',
     &//,'    FLUX ',
     &   ' to generate the particle'
     &   ' environment outside the spacecraft; and'
     & /,'    TRANS',
     &   ' to transport fluxes through the spacecraft shielding.',
     &//,' NOTE: Before running this or any other CREME96 programs',
     &   ' please define three ',
     & /,' logicals: ',/,
     & /,4x,' CREME96       as the directory where CREME96 source',
     &        ' & executables reside.',
     & /,4x,' CR96TABLES    as the directory in which CREME96 data',
     &        ' tables reside.',
     & /,4x,' USER          as the directory in which output files',
     &        ' should be written.',
     &  //,' Now begin specification of the LETSPEC_DRIVER inputs: ')

      INFILE='      '

  112 CONTINUE
```

```
      CALL RETRY_INPUT(IERR)
      WRITE(6,1002)
 1002 FORMAT(' Enter name of file containing',
     &         ' CREME96 particle fluxes, ie. something.TFX',
     &       /,' from TRANS or',
     &         ' (for zero shielding) something.FLX from FLUX'
     &       /,' or something.tr* from TRP:')
      READ(*,1014,ERR=112,IOSTAT=IERR) INFILE
 1014 FORMAT(A80)

      IF (INFILE.EQ.IBLANK) THEN
          WRITE(6,1914)
 1914     FORMAT(1x,' You must specify an input .FLX file here,',
     &              ' either from TRANS or',
     &    /,1x,' (in the case of zero shielding) from',
     &         ' FLUX. ',
     &              ' Please try again.',/)
           GOTO 112
      ELSE
          IFILETYPE=4
          WRITE(6,1020) INFILE
 1020     FORMAT(' Input Flux File =',/,1x,A80)
          CALL CHECK_FILE(IFILETYPE,INFILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 112
      ENDIF

  103 CONTINUE
C
C     Modification 9/12/96: LET range hardwired:
          LETMINMG=1.0E-3
          LETMAXMG=1.1E+2
C-----------------------------------------------------------------------------
C     Following code obsolete 9/12/96:
C
C      WRITE(6,1003)
C 1003 FORMAT(/,' Enter minimum & maximum LET values (in MeV-cm2/mg) ',
C    &        /,' [Recommended for most SEE applications:'
C    &        /,'      minimum LET = 1.0E-3 MeV-cm2/mg  ',
C    &        /,'      maximum LET = 1.1E+2 MeV-cm2/mg]  ',
C    &        /,' NOTE THE UNITS USED HERE -- per milligram!',
C    &  /,' Enter 0 0 <CARRIAGE RETURN> for recommended defaults.)')
C      ACCEPT *, LETMINTEMP,LETMAXTEMP
C      LETMINMG=MIN(LETMINTEMP,LETMAXTEMP)
C      LETMAXMG=MAX(LETMINTEMP,LETMAXTEMP)
C      IF ((LETMINMG.EQ.0. .and. LETMAXMG.EQ.0.)
C    &     .or. (LETMAXMG.LE.LETMINMG) ) THEN
C          LETMINMG=1.0E-3
C          LETMAXMG=1.1E+2
C      ENDIF
C      IF (LETMINMG.LT.0. .or. LETMAXMG.LT.0.
C    &     .or.LETMINMG.EQ.LETMAXMG) THEN
C         WRITE(6,9001) LETMINMG,LETMAXMG
C 9001    FORMAT(1x,' Invalid LET value(s): ',E13.6,2x,E13.6,
C    &         /,1x,' Please try again.')
C         GOTO 103
C      ENDIF

CC      WRITE(6,1031) LETMINMG,LETMAXMG
 1031 FORMAT(/,' Integral LET spectrum accumulated for ',
     &       /,1x,E12.5,' </= LET </= ',E12.5,' MeV-cm2/mg',/)
```

```
 1032  CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,1004)
 1004  FORMAT(' Enter minimum & maximum atomic numbers to be',
     &           ' included in integral LET spectrum:',
     &         /,'    [Enter 0 0 <CARRIAGE RETURN> for full range',
     &           ' included in the input flux file.',
     &         /,'      NOTE: in general',
     &           ' protons (Z=1) should NOT be included in the',
     &           ' LET spectrum',/,'    for most SEU calculations.]')

 101   CONTINUE
       READ(*,*,ERR=1032,IOSTAT=IERR) IMINTEMP,IMAXTEMP

       IF (IMAXTEMP.NE.0 .and. IMAXTEMP.LT.IMINTEMP) THEN
           IZMIN=MIN(IMINTEMP,IMAXTEMP)
           IZMAX=MAX(IMINTEMP,IMAXTEMP)
       ELSE
           IZMIN=IMINTEMP
           IZMAX=IMAXTEMP
       ENDIF

       IF (IZMIN.LT.0 .or. IZMIN.GT.92
     &     .or. IZMAX.LT.0 .or. IZMAX.GT.92) THEN
               WRITE(6,9002) IZMIN,IZMAX
 9002         FORMAT(1x,' Invalid atomic number(s): ',2I5,
     &               /,1x,' Please try again.')
               GOTO 101
       ENDIF

       IF (IZMIN.EQ.0 .and. IZMAX.EQ.0) THEN
           WRITE(6,1039)
 1039  FORMAT(' Nominal Z range from input flux file used.',/)
       ELSEIF (IZMIN.EQ.0 .and. IZMAX.NE.0) THEN
           WRITE(6,1038) IZMAX
 1038  FORMAT(' Minimum Z value from input flux file; Maximum Z =',I3)
       ELSEIF (IZMIN.NE.0 .and. IZMAX.EQ.0) THEN
           WRITE(6,1037) IZMIN
 1037  FORMAT(' Minimum Z =',I3,'; Maximum Z value from',
     &        ' input flux file.')
       ELSE
           WRITE(6,1040) IZMIN,IZMAX
 1040  FORMAT(1x,' Integral LET spectrum accumulated for elements',
     &      /,1x,I3,' </= Z </= ',I3,/)
       ENDIF

  105  CONTINUE
       CALL RETRY_INPUT(IERR)
       EMAXCUT=1.0E+24
       WRITE(6,1005)
 1005  FORMAT(' Enter minimum particle energy (in MeV/nuc)',
     &          ' to be included in accumulating the'
     &        /,' integral LET spectrum:',
     &     /,'    [NOTE: for most SEU applications,',
     &       ' the recommended value = 0.1 MeV/nuc.',
     &     /,'     However, in some devices, ranging out of low-energy',
     &       ' particles along very',
     &     /,'     long RPP chords can lead to gross overestimates,',
     &       ' particularly for low-',
```

```
     &    /,'    threshold devices in solar particle events.  In',
     &      ' these cases, larger minimum',
     &    /,'    energy values (1-10 MeV/nuc) should be explored.]')

        READ(*,*,ERR=105,IOSTAT=IERR) EMINCUT
        IF (EMINCUT.LT.0.) THEN
           WRITE(6,9005) EMINCUT
9005       FORMAT(1x,' Invalid minimum energy value: ',E13.6,
     &          /,1x,' Please try again.')
           GOTO 105
        ENDIF

      WRITE(6,1051) EMINCUT
1051  FORMAT(/,' Integral LET spectrum accumulated for',
     &          ' nuclei with energy > ',F8.3,' MeV/nuc.')


      MATERIAL='SILICON'
      WRITE(6,1007) MATERIAL
1007  FORMAT(/,' LET spectrum calculated in ',A12)

1017  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1008)
1008  FORMAT(/,' Enter name of output file: ',
     &        /,' Note: for standard input to CREME96 SEU routines',
     &          ' must be something.LET')

      ILONG=INDEX(INFILE,'.')
      IF (ILONG.NE.0) THEN
          DEFAULT_NAME=INFILE(1:ILONG)//'LET'
      ELSE
          DEFAULT_NAME=INFILE//'.LET'
      ENDIF

      WRITE(6,1028) DEFAULT_NAME(1:79)
1028  FORMAT(' Suggested name:',/,1x,A79,
     &       /,' Hit RETURN if this is acceptable.')

1018  CONTINUE

      READ(*,1014,ERR=1017,IOSTAT=IERR) OUTFILE
      IF (OUTFILE.EQ.IBLANK) OUTFILE=DEFAULT_NAME

      WRITE(6,1009) OUTFILE
1009  FORMAT(' Output Flux File =',/,1x,A80)

      CALL CHECK_NAME_CONFLICT(INFILE,OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) GOTO 1017

      CALL CHECK_OUTPUT_FILE(OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) THEN
      WRITE(6,1010) INFILE(1:75)
      WRITE(6,1011) OUTFILE(1:75)
      WRITE(6,1012)
1010  FORMAT(1x,' INPUT file = ',/,5x,A75)
1011  FORMAT(1x,' Previous try at OUTPUT name = ',/,5x,A75)
1012  FORMAT(1x,' Try again, ie., newname.LET')
      GOTO 1018
      ENDIF
```

```
1060  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1061)
1061  FORMAT(/,' Do you want a DIFFERENTIAL LET spectrum also?: ',
     &          ' (0=no; 1=yes) ',
     &      /,'   NOTE: A differential LET spectrum is NOT necessary',
     &        ' for SEU calculations.')

      READ(*,*,ERR=1060,IOSTAT=IERR)  IDIFSPEC
      IF (IDIFSPEC.NE.1) IDIFSPEC=0
      IF (IDIFSPEC.EQ.0) WRITE(6,1062)
      IF (IDIFSPEC.EQ.1) WRITE(6,1063)
1062  FORMAT(' No differential LET spectrum will be created.',/)
1063  FORMAT(' Differential LET spectrum also created. The file name',
     &        ' will be the same as',
     &      /,' that of the integral LET spectrum,'
     &        ' but with extension .DLT',/)


      RETURN
      END
```

```
      SUBROUTINE INIPROP(INFILE,IPATH,UPATH,TARGET,
     &                    SHIELDFILE,OUTFILE)

C
C     Subroutine for initializing input parameters to transport routine
C     in CREME96.  This version only allows ALUMINUM shielding.
C
C     Modified 06-13-96: to include shielding distribution
C     Modified 11-13-96: gets shielding distribution from standard *.SHD
C                        file, as created with the SHIELDFILE_DRIVER program.
C     Modified 11-17-97: to include .trp inputs
C
      IMPLICIT NONE
      INTEGER*4 IPATH,IULABEL,IFILETYPE,IACCEPT
      INTEGER*4 KFILETYPE,KACCEPT
      REAL*4 UPATH
      CHARACTER*1 IBLANK
      DATA IBLANK/' '/
      CHARACTER*80 INFILE,SHIELDFILE,OUTFILE
      CHARACTER*12 TARGET

      CHARACTER*5 UNITS_LABEL
      DIMENSION UNITS_LABEL(4)
      DATA UNITS_LABEL/'g/cm2','mils ','cm   ','!!!!!'/

      INTEGER*4 IERR
      DATA IERR/0/
C
      WRITE(6,1000)
 1000 FORMAT(' CREME96 IONIZING RADIATION ENVIRONMENT MODEL',
     &     /,' -->  NUCLEAR TRANSPORT PROGRAM')
      WRITE(6,1001)
 1001 FORMAT(' This program will transport the ionizing',
     &       '-radiation particle fluxes generated by',
     &     /,' the CREME96 code through aluminum shielding',
     &       ' of specified thickness.  Before',
     &     /,' running this program, you must do FLUX',
     &       ' (ie, run CREME96:FLUX_DRIVER)',
     &     /,'  or TRP',
     &       ' (ie, run CREME96:TRAPPED_PROTON_DRIVER)',
     &       ' to generate',/,' the particle environment',
     &       ' outside of the spacecraft.  After running',
     &       ' this program',/,' you will run other routines to',
     &       ' calculate SEU rates. ',
     &//,' NOTE: Before running this or any other CREME96 programs',
     &  ' please define three ',
     & /,' logicals: ',/,
     & /,4x,' CREME96       as the directory where CREME96 source',
     &       ' & executables reside.',
     & /,4x,' CR96TABLES    as the directory in which CREME96 data',
     &       ' tables reside.',
     & /,4x,' USER          as the directory in which output files',
     &       ' should be written.',
     &  //,' Now begin specification of the transport parameters: ')

       INFILE='        '

  112  CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,1002)
```

```
1002  FORMAT(/,' Enter name of file containing',
     &          ' CREME96 particle fluxes:',
     &        /,' ie., something.FLX, something.TRP, or something.TFX')
      READ(*,1014,ERR=112,IOSTAT=IERR) INFILE
1014  FORMAT(A80)
      IF (INFILE.EQ.IBLANK) THEN
          WRITE(6,1914)
1914      FORMAT(1x,' You must specify here EITHER a .FLX file',
     &                  ' from FLUX (FLUX_DRIVER)',
     &             /,1x,' OR a .TR* file from a ',
     &                  ' TRP (TRAPPED_PROTON_DRIVER). ',
     &             /,1x,' OR a .TFX file from a previous run of',
     &                  ' TRANS (TRANSPORT_DRIVER). ',
     &             /,1x,' Please try again.',/)
           GOTO 112
      ELSE
          IFILETYPE=3
          WRITE(6,1020) INFILE
1020      FORMAT(' Input Flux File =',/,1x,A80)
          CALL CHECK_FILE(IFILETYPE,INFILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 112
      ENDIF

      TARGET='ALUMINUM'

1021  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1003) TARGET(1:8)
1003  FORMAT(/,' In what units will the ',A8,
     &          ' shielding thickness be given? ',
     &        /,' Enter 0, 1, or 2: ',
     &        /,'                       0 = g/cm**2',
     &        /,'                       1 = mils   ',
     &        /,'                       2 = cm     ',
     &       //,' (Note: 100 mils = 0.254 cm = 0.6858 g/cm**2 Al.)')

      READ(*,*,ERR=1021,IOSTAT=IERR) IPATH
      IF (IPATH.LT.0 .or. IPATH.GT.2) THEN
          WRITE(6,9000)
9000      FORMAT(1x,' Illegal units specification.  Please try again.')
          GOTO 1021
      ENDIF
      IULABEL=IPATH+1
      IF (IULABEL.GT.4) IULABEL=4
      WRITE(6,9001) UNITS_LABEL(IULABEL),TARGET
9001  FORMAT(1x,' Shielding thicknesses: in ',A5,1x,A12)

      SHIELDFILE='            '

      WRITE(6,1035)
1035  FORMAT(/,1x,' COMMENT ON SHIELDING VALUES:  It is common',
     &            ' practice for researchers dealing',
     &        /,1x,' with total dose and dose-rate effects to',
     &            ' determine part response with zero',
     &        /,1x,' shielding.  For single event effects, on the '
     &            ' other hand, it is important to',
     &        /,1x,' shield out low-energy particles',
     &            ' which would never be encountered in a',
     &        /,1x,' realistic situation.  A nominal shielding',
     &              ' thickness of 100 mils is therefore',
```

```
     &        /,1x,' recommended for general comparison purposes. ',
     &             ' However, a realistic',
     &             ' shielding',
     &        /,1x,' distribution is essential for accurate SEU',
     &             ' calculations in solar energetic',
     &        /,1x,' particle ("flare") environments.')
 1036 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1004)
 1004 FORMAT(/,1x,' Enter shielding thickness: ',
     &        /,1x' [Enter 0 if you wish to specify a file'
     &            ' containing a shielding distribution.]')

      READ(*,*,ERR=1036,IOSTAT=IERR) UPATH

      IF (UPATH.GT.0.0) THEN
          WRITE(6,1005) UPATH,UNITS_LABEL(IULABEL),TARGET
 1005     FORMAT(' Shielding thickness = ',F10.5,1x,A5,5x,A12)
      ELSE
 1039    CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,1041)
 1041 FORMAT(/,1x,' Enter name of file containing shielding',
     &           ' distribution.  This file, which should',
     &      /,1x,' be called something.SHD, has a special',
     &           ' format.  This file must have been',
     &      /,1x,' created before running TRANS with the',
     &           ' CREME96 command SHIELDFILE.')

         READ(*,1014,ERR=1039,IOSTAT=IERR) SHIELDFILE
         WRITE(6,1042) SHIELDFILE
 1042    FORMAT(' Shielding File =',/,1x,A80)
         KACCEPT=0
C        KFILETYPE=0
         KFILETYPE=7
         CALL CHECK_FILE(KFILETYPE,SHIELDFILE,KACCEPT)
         IF (KACCEPT.NE.0) GOTO 1039
      ENDIF


      WRITE(6,1008)
 1008 FORMAT(/,' Enter name of output file, ie., newname.TFX:')

 1043 CONTINUE
      CALL RETRY_INPUT(IERR)
      READ(*,1014,ERR=1043,IOSTAT=IERR) OUTFILE
      WRITE(6,1009) OUTFILE
 1009 FORMAT(' Output Flux File =',/,1x,A80)

      CALL CHECK_OUTPUT_FILE(OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) THEN
      WRITE(6,1010) INFILE(1:75)
      WRITE(6,1011) OUTFILE(1:75)
      WRITE(6,1012)
 1010 FORMAT(1x,' INPUT file = ',/,5x,A75)
 1011 FORMAT(1x,' Previous try at OUTPUT name = ',/,5x,A75)
 1012 FORMAT(1x,' Try again, ie., newname.TFX')
      GOTO 1043
      ENDIF
```

```
RETURN

END
```

```
      SUBROUTINE INISHIELD(MAXSHIELD,COMMENT,
     &                     IUNITS,MATERIAL,NBINS,XTHICK,XPROB,
     &                     SHIELDFILE)

C
C     Subroutine for initializing inputs to CREME96 program which
C     creates a shielding distribution file in standard format.
C     This version only allows ALUMINUM shielding.
C
      IMPLICIT NONE
      INTEGER*4 IUNITS,NBINS,MAXSHIELD,IULABEL
      CHARACTER*12 MATERIAL
      CHARACTER* 1 IBLANK
      DATA IBLANK/' '/
      REAL*4 XTHICK(1),XPROB(1)
      REAL*4 XTEMP,PTEMP
      CHARACTER*80 COMMENT,SHIELDFILE
      CHARACTER*5 UNITS_LABEL
      DIMENSION UNITS_LABEL(4)
      DATA UNITS_LABEL/'g/cm2','mils ','cm   ','!!!!!'/
C
      INTEGER*4 IERR,IACCEPT
      DATA IERR/0/
C
      WRITE(6,1000)
 1000 FORMAT(' CREME96 IONIZING RADIATION ENVIRONMENT MODEL',
     &     /,' -->  Shielding distribution program')
      WRITE(6,1001)
 1001 FORMAT(' This auxilliary program will use',
     &        ' specified inputs to create a shielding '
     &     /,' distribution file with the format',
     &        ' and header information expected by CREME96.',
     &//,' NOTE: Before running this program please define',
     &        ' three logicals:',/,
     & /,4x,' CREME96       as the directory where CREME96 source',
     &        ' & executables reside.',
     & /,4x,' CR96TABLES    as the directory in which CREME96 data',
     &        ' tables reside.',
     & /,4x,' USER          as the directory in which output files',
     &        ' should be written.',
     &  //,' Now begin specification of the shieldfile inputs: ')


  105  CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,1215)
 1215  FORMAT(/,1x,' Enter comment (80 characters max)',
     &              ' for record-keeping in output file:')
       READ(*,1218,ERR=105,IOSTAT=IERR) COMMENT
 1218  FORMAT(A80)

 1021 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1003)
 1003 FORMAT(/,' In what units will the shielding thickness be given? ',
     &          ' Enter 0, 1, or 2: ',
     &        /,'      0 = g/cm**2',
     &        /,'      1 = mils   ',
     &        /,'      2 = cm     ',
     &        /,' (Note: 100 mils = 0.254 cm = 0.6858 g/cm**2 Al.)')
```

```
      READ(*,*,ERR=1021,IOSTAT=IERR) IUNITS
      IF (IUNITS.LT.0 .or. IUNITS.GT.2) THEN
          WRITE(6,9003)
 9003     FORMAT(1x,' Illegal units specification.  Please try again.')
          GOTO 1021
      ENDIF

      IULABEL=IUNITS+1
      IF (IULABEL.GT.4) IULABEL=4
      MATERIAL='ALUMINUM'


      WRITE(6,9000) UNITS_LABEL(IULABEL),MATERIAL
 9000 FORMAT(1x,' Shielding thickness in',A5,2x,A12)


      NBINS=0
      WRITE(6,1041) MAXSHIELD
 1041 FORMAT(/,1x,' Now begin entry of shielding distribution: ',
     & /,1x,' Enter thickness (in the units specified above) and',
     &      ' coverage factor (a number',
     & /,1x,' between 0 and 1) then <CARRIAGE RETURN>',
     &      ' for each bin of the distribution.',
     & /,1x,' Terminate your input list with 0 0 <CARRIAGE RETURN>.',
     & /,1x,' The maximum number of bins allowed in the',
     &      ' distribution is',I4,'.')
 1036 CONTINUE
      WRITE(6,1042)
 1042 FORMAT(/,1x,' Enter shielding thickness and coverage factor: ')
      CALL RETRY_INPUT(IERR)
      READ(*,*,ERR=1036,IOSTAT=IERR) XTEMP,PTEMP
      IF (XTEMP.GT.0.0 .and. PTEMP.GT.0.0) THEN
           NBINS=NBINS+1
           IF (NBINS.LE.MAXSHIELD) THEN
               XTHICK(NBINS)=XTEMP
               XPROB(NBINS)=PTEMP
               WRITE(6,999) NBINS,XTHICK(NBINS),UNITS_LABEL(IULABEL),
     &                      XPROB(NBINS)
 999           FORMAT(1x,' SHIELDING BIN ',I4,' THICKNESS = ',
     &                     F10.4,1x,A5,'   FRACTION = ',F8.4)
           ELSE
               WRITE(6,1043) NBINS,MAXSHIELD
 1043          FORMAT(1x,' Input terminated: No. input bins  = ',I5,
     &                /,1x,'                    Maximum allowed = ',I5)
               GOTO 1050
           ENDIF
      ELSE
           WRITE(6,1044)
 1044      FORMAT(1x,' Shielding distribution input completed.')
           GOTO 1050
      ENDIF
      GOTO 1036

 1050 CONTINUE

      SHIELDFILE='        '

  112 CONTINUE
      CALL RETRY_INPUT(IERR)
```

```
      WRITE(6,1002)
1002  FORMAT(/,' Enter name of output shielding file:',
     &          ' ie., something.SHD  (Your file must have',
     &        /,' this extension in order',
     &          ' to be accessbile by CREME96',
     &          ' directory routines.)')
      READ(*,1014,ERR=112,IOSTAT=IERR) SHIELDFILE
1014  FORMAT(A80)
      IF (SHIELDFILE.EQ.IBLANK) THEN
          WRITE(6,1914)
1914      FORMAT(1x,' You must specify a filename here:',
     &           /,1x,' Please try again.',/)
           GOTO 112
      ELSE
          WRITE(6,1020) ShieldFILE
1020      FORMAT(' Output Shielding Distribution File =',/,1x,A80)
          CALL CHECK_OUTPUT_FILE(SHIELDFILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 112
      ENDIF

      RETURN

      END
```

```
      SUBROUTINE INITIALIZE_HI_UPSETS(LET_FILE,XM,YM,ZM,FUNNEL,NBITS,
     &                              IPARAM,PARAMS,XSECT_FILE,IENTER,
     &                              DEVICE_LABEL,REPORT_FILE)
C
C
C     Generates interactive dialogue to get necessary input parameters
C     for heavy-ion upsets:
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C
C-----------------------------------------------------------------------
C
      IMPLICIT NONE
      CHARACTER*80 LET_FILE,XSECT_FILE,REPORT_FILE
      CHARACTER*40 DEVICE_LABEL
      REAL*4 XM,YM,ZM,FUNNEL,PARAMS,NBITS
      INTEGER*4 IPARAM,IACCEPT,IFILETYPE,IENTER
      CHARACTER*1 IBLANK
      DATA IBLANK/' '/
      DIMENSION PARAMS(4)
      INTEGER*4 IERR
      DATA IERR/0/

      IF (IENTER.EQ.1) THEN

      WRITE(6,1000)
 1000 FORMAT(' CREME96 IONIZING RADIATION ENVIRONMENT MODEL',
     & /,' -->  Heavy-Ion-Induced Single Event Upset',
     &   ' (SEU) Rate Calculation')
      WRITE(6,1001)
 1001 FORMAT(' This program will use the integral LET ',
     &       ' spectrum (something.LET,  generated by ',
     &      /,' the CREME96 codes) and device'
     &       ' characteristics (input below) to calculate',
     &      /,' a heavy-ion induced SEU rate (in upsets/bit'
     &       ' /sec or /day).',
     &      /,' Before running this program you must do:',
     &  /,'   FLUX      to generate the environment',
     &      ' outside of the spacecraft;',
     &  /,'   TRANS     to transport the particle fluxes',
     &     ' through shielding; and',
     &  /,'   LETSPEC   to create an integral LET spectrum.'
     &    /,' For many devices and applications you should also do:',
     &  /,'   PUP       to calculate the rate',
     &     ' of proton-induced SEUs.',
     &//,' NOTE: Before running this or any other CREME96 programs',
     &   ' please define three ',
     & /,' logicals: ',/,
     & /,4x,' CREME96       as the directory where CREME96 source',
     &       ' & executables reside.',
     & /,4x,' CR96TABLES    as the directory in which CREME96 data',
     &       ' tables reside.',
     & /,4x,' USER          as the directory in which output files',
     &       ' should be written.',
     &  //,' Now begin specification of inputs for the',
```

```
    &     ' SEU rate calculation: ')

       LET_FILE='         '

  110  CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,1100)
 1100  FORMAT(' Enter name of integral LET spectrum file',
    &          ' (something.LET):')
       READ(*,1,ERR=110,IOSTAT=IERR) LET_FILE
 1     FORMAT(A80)

      IF (LET_FILE.EQ.IBLANK) THEN
          WRITE(6,1914)
 1914     FORMAT(1x,' You must specify an input .LET file',
    &               ' from LETSPEC_DRIVER here. ',
    &               ' Please try again.',/)
           CALL CHECK_FILE(IFILETYPE,LET_FILE,IACCEPT)
           GOTO 110
      ELSE
          IFILETYPE=5
          WRITE(6,1110) LET_FILE
 1110     FORMAT(' Input LET File = ',/,1x,A80)
          CALL CHECK_FILE(IFILETYPE,LET_FILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 110
      ENDIF


  120 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1120)
 1120 FORMAT(1x,' Enter name for an output file, which will',
    &    ' record the inputs and results. ',
    & /,1x,' (If no report file is wanted, hit <CARRIAGE RETURN>.)')
  121 CONTINUE
      READ(*,1,ERR=120,IOSTAT=IERR) REPORT_FILE

      IF (REPORT_FILE(1:2).EQ.'-1') GOTO 110

      IF (REPORT_FILE.EQ.IBLANK) THEN
          REPORT_FILE='NULLFILE'
          WRITE(6,1121)
 1121     FORMAT(1x,' No report file created by HI_UPSET_DRIVER.')
      ELSE
          CALL CHECK_OUTPUT_FILE(REPORT_FILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 121
          WRITE(6,1122) REPORT_FILE(1:79)
 1122     FORMAT(1x,' Report file created by HI_UPSET_DRIVER: ',
    &          /,1x,A79)
      ENDIF

      ENDIF

      CALL GET_HI_XS_INPUTS(DEVICE_LABEL,XM,YM,ZM,FUNNEL,NBITS,
    &                       IPARAM,PARAMS,XSECT_FILE)

      IF (IENTER.EQ.1 .and. DEVICE_LABEL(1:2).EQ.'-1') GOTO 120

      RETURN
      END
```

```
      SUBROUTINE INITIALIZE_PROTON_UPSETS(PROTON_FILE,NBITS,
     &                                   IPARAM,PARAMS,XSECT_FILE,
     &                                   IENTER,
     &                                   DEVICE_LABEL,REPORT_FILE)
C
C
C     Generates interactive dialogue to get necessary input parameters
C     for proton-induced SEU rate:
C
C     Written by:   Allan J. Tylka
C                   Code 7654
C                   Naval Research Laboratory
C                   Washington, DC 20375-5352
C                   tylka@crs2.nrl.navy.mil
C
C     Last update:  20 August 1996
C
C-----------------------------------------------------------------------
C
      IMPLICIT NONE
      CHARACTER*80 PROTON_FILE,XSECT_FILE,REPORT_FILE
      CHARACTER*40 DEVICE_LABEL
      REAL*4 PARAMS,NBITS
      INTEGER*4 IPARAM,IENTER,IFILETYPE,IACCEPT
      DIMENSION PARAMS(4)
      CHARACTER*1 IBLANK
      DATA IBLANK/' '/
      INTEGER*4 IERR
      DATA IERR/0/

     IF (IENTER.EQ.1) THEN

     WRITE(6,1000)
1000 FORMAT(' CREME96 IONIZING RADIATION ENVIRONMENT MODEL',
    & /,' -->  Proton-Induced Single Event Upset',
    &   ' (SEU) Rate Calculation')
     WRITE(6,1001)
1001 FORMAT(' This program will use the differential',
    &        ' proton flux generated by CREME96',
    &      /,' [something.TFX or (for zero shielding) something.FLX',
    &        ' or .tr*]',
    &        ' and device characteristics',
    &      /,' [input below] to calculate a proton-induced',
    &        ' SEU rate (in SEUs/bit/sec or /day).',
    &      /,' NOTE: the .TFX/.FLX file may contain other species',
    &        ' in addition to protons, but',
    &      /,' they will be ignored here.',
    &     //,' Before running this program, you must do:',
    &      /,'    FLUX  to generate the particle',
    &        ' environment outside of the spacecraft;',
    &      /,'    TRANS to transport the',
    &        ' particle fluxes through shielding.',
    &      /,' For many devices and applications',
    &        ' you should also do: ',
    &      /,'    LETSPEC and HUP to calculate the rate of',
    &       ' heavy-ion-induced SEUs.',
    &//,' NOTE: Before running this or any other CREME96 programs',
    &   ' please define three ',
    & /,' logicals: ',/,
    & /,4x,' CREME96       as the directory where CREME96 source',
```

```
     &        ' & executables reside.',
     & /,4x,' CR96TABLES    as the directory in which CREME96 data',
     &        ' tables reside.',
     & /,4x,' USER          as the directory in which output files',
     &        ' should be written.',
     & //,' Now begin specification of inputs for the',
     &     ' proton-SEU rate calculation: ')


110    CONTINUE
       CALL RETRY_INPUT(IERR)
       WRITE(6,1100)
1100   FORMAT(' Enter name of flux file containing the proton spectrum',
     &         ' (something.TFX or .FLX):')
       READ(*,1,ERR=110,IOSTAT=IERR) PROTON_FILE
1      FORMAT(A80)
       IF (PROTON_FILE.EQ.IBLANK) THEN
          WRITE(6,1914)
1914      FORMAT(1x,' You must specify either an input .TFX file',
     &              ' from TRANSPORT_DRIVER or',
     &    /,1x,' (in the case of zero shielding) an input',
     &              ' .FLX file from FLUX_DRIVER',
     &    /,1x,' Please try again.',/)
           GOTO 110
      ELSE
          IFILETYPE=4
          WRITE(6,1110) PROTON_FILE
1110      FORMAT(' Input Particle Flux File (containing protons) = '
     &           ,/,1x,A80)
          CALL CHECK_FILE(IFILETYPE,PROTON_FILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 110
      ENDIF

 120  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,1120)
1120  FORMAT(1x,' Enter name for an output file, which will',
     &    ' record the inputs and results. ',
     & /,1x,' (If no report file is wanted, hit <CARRIAGE RETURN>.)')
 121  CONTINUE
      READ(*,1,ERR=120,IOSTAT=IERR) REPORT_FILE

      IF (REPORT_FILE(1:2).EQ.'-1') GOTO 110

      IF (REPORT_FILE.EQ.IBLANK) THEN
          REPORT_FILE='NULLFILE'
          WRITE(6,1121)
1121      FORMAT(1x,' No report file created by HI_UPSET_DRIVER.')
      ELSE
          CALL CHECK_OUTPUT_FILE(REPORT_FILE,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 121
          WRITE(6,1122) REPORT_FILE(1:79)
1122      FORMAT(1x,' Report file created by HI_UPSET_DRIVER: ',
     &          /,1x,A79)
      ENDIF


      ENDIF

      CALL GET_PROTON_XS_INPUTS(DEVICE_LABEL,
```

```
     &                              NBITS,IPARAM,PARAMS,XSECT_FILE)
      IF (IENTER.EQ.1 .and. DEVICE_LABEL(1:2).EQ.'-1') GOTO 120

      RETURN
      END
```

```
      SUBROUTINE INTEGRATE_HEAVY_ION_UPSETS(NPTS,LETG,FLUX,XSECT,
     &                                      XM,YM,ZM,FUNNELM,
     &                                      SEU_RATE)
C
C     Subroutine for performing numerical integration over the soft-turn
C     on in the heavy-ion LET-dependent cross-section
C
C     INPUTS:        NPTS  = number of datapoints in input array
C                    LETG  = array containing LET values in MeV-cm2/g
C                    FLUX  = array containing the flux (in /m2-s-sr)
C                            of particles with LET > LETG
C                    XSECT = array of cross-section values (in 1.0E-8 cm2/bit)
C                            corresponding to the LETG values
C                    XM,YM,ZM = device dimensions in microns
C                               (ZM = depth of sensitive area,
C                                     typically 0.5-2.0 microns)
C                    FUNNELM = funnel length (in microns)
C
C     OUTPUT:        SEU_RATE = SEU rate in SEUS/s/bit
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C     Last update:   24 October 1996
C
C-----------------------------------------------------------------------------
C
      IMPLICIT NONE
      INTEGER*4 NPTS,NPTSMAX,K
      PARAMETER(NPTSMAX=5000)
      REAL*4 LETG,FLUX,XSECT,XM,YM,ZM,FUNNELM,SEU_RATE,QC,AFRAC,
     &       UPS1,UPS2,DELTA_U,DELTA_U_SAVE,AFRACMAX
      DATA AFRACMAX/0.99/
      DIMENSION LETG(1),FLUX(1),XSECT(1)
      DIMENSION QC(NPTSMAX),AFRAC(NPTSMAX)
      INTEGER*4 NERRORS,QPTS,KLAST
      LOGICAL QUIET
      DATA QUIET/.true./
C
C     Integrates over the soft turn-on in the heavy-ion SEU cross-section
C     by calculating rate at each critical charge value and adding
C     together in a sum weighted by the corresponding cross-section
C
      CALL SAMPLE_SOFT_TURN_ON(NPTS,NPTSMAX,
     &                         LETG,XSECT,XM,YM,ZM,FUNNELM,
     &                         AFRACMAX,
     &                         QPTS,QC,AFRAC)


C
C     Now calculate upsets for each critical charge interval
      NERRORS=0
      SEU_RATE=0.0

      CALL GET_UPSET(XM,YM,ZM,FUNNELM,QC(1),NPTS,LETG,FLUX,UPS1)
      DO 200 K=1,QPTS-1
```

```
          CALL GET_UPSET(XM,YM,ZM,FUNNELM,QC(K+1),NPTS,LETG,FLUX,UPS2)
          DELTA_U=(UPS1-UPS2)*0.5*(AFRAC(K)+AFRAC(K+1))

          IF (DELTA_U.LT.0.) THEN
C
C         10/24/96: The original versions of CREME and CREME96 contained
C         an occasional bug here. The SEU rate should, of course,
C         be a monotonically decreasing function of increasing critical
C         charge. Due to poor sampling around LETs corresponding to the peaks
C         in the differential pathlength distribution, occasionally
C         the rates returned from GET_UPSET did not show this. This problem
C         has now (I believe) been fixed, by explicitly installing a
C         higher density sampling around these values. The following
C         error message should therefore NEVER be activated.  AJT

          NERRORS=NERRORS+1
          IF (.not. quiet) WRITE(6,9999) DELTA_U,K,  QC(K),  UPS1,
     &                                   K+1,QC(K+1),UPS2,
     &                                   AFRAC(K),AFRAC(K+1)
 9999     FORMAT(1x,'  ERROR in SEU vs. QC: DELTA_U = ',E13.6,
     &         /,1x,'  K   ,QC(K)   ,UPS1: ', I6,1x,E13.6,5x,E13.6,
     &         /,1x,'  K+1,QC(K+1),UPS2: ', I6,1x,E13.6,5x,E13.6,
     &         /,1x,'  AFRAC(K),AFRAC(K+1) : ',E13.6, 5x,E13.6,
     &         /,1x,'  PLEASE NOTIFY tylka@crs2.nrl.navy.mil.')

          ELSE

          DELTA_U_SAVE=DELTA_U

          ENDIF

C         Protect against bug of undetermined origin:
          IF (DELTA_U.LT.0.) DELTA_U=DELTA_U_SAVE
          SEU_RATE=SEU_RATE+DELTA_U


          IF (AFRAC(K+1).GE.AFRACMAX) THEN
C             Plateau in cross-section reached.  Terminate numerical
C             integration over soft turn-on.
              KLAST=K+1
              GOTO 210
          ENDIF

C         Store SEU rate for next integration step:
          UPS1=UPS2

 200    CONTINUE
        KLAST=QPTS

 210    CONTINUE

        CALL GET_UPSET(XM,YM,ZM,FUNNELM,QC(KLAST),NPTS,LETG,FLUX,UPS1)
        IF (UPS1.LT.0.) UPS1=0.0
        SEU_RATE=SEU_RATE+UPS1*AFRAC(KLAST)
C
        IF (.not.quiet.and.NERRORS.GT.0.)WRITE(6,9990) QPTS,NERRORS
 9990   FORMAT(1X,' Debug I_H_I_U: NPTS,NERRORS = ',2I5)
        RETURN
        END
```

```
      SUBROUTINE SAMPLE_SOFT_TURN_ON(NPTS,NPTSMAX,
     &                               LETG,XSECT,XM,YM,ZM,FUNNELM,
     &                               AFRACMAX,
     &                               QPTS,QC,AFRAC)

      IMPLICIT NONE
      INTEGER*4 NPTS,NPTSMAX,QPTS
      REAL*4 LETG,XSECT,XM,YM,ZM,FUNNELM,AFRACMAX,QC,AFRAC
      DIMENSION LETG(1),XSECT(1),QC(1),AFRAC(1)
      INTEGER*4 K,NSAMPMAX,KMOD,KLAST0,KFIRST,KLAST
      REAL*4 AFRACTEST
      DATA NSAMPMAX/100/

C     Comparison of SEU rates for various devices showed that sampling
C     the turn on region of the cross-section curve in 100-200 points
C     should be adequate.  NSAMPMAX is used to reduce the number of
C     points in the cross-section sampling to this level.
C
      QPTS=1
      DO 50 K=2,NPTS
         IF (XSECT(K).GT.0.) THEN
             QPTS=QPTS+1
             KLAST0=K
             AFRACTEST=XSECT(K)/(XM*YM)
             IF (AFRACTEST.GE.AFRACMAX) GOTO 51
         ENDIF
 50   CONTINUE
 51   CONTINUE
      KMOD=QPTS/NSAMPMAX
      IF (KMOD.LE.1) KMOD=1

      QPTS=1
      KFIRST=1
      DO 100 K=2,NPTS,KMOD
         IF (XSECT(K).GT.0.) THEN
             IF (QPTS.EQ.1) KFIRST=K-1
C            Now convert LET values in cross-section table to critical
C            charges, in picocoulombs.  See SEE Notebook #1, p. 5
             QPTS=QPTS+1
             QC(QPTS)=LETG(K)*(ZM+FUNNELM)*1.033E-5
C            Also scale cross-section value by nominal area:
             AFRAC(QPTS)=XSECT(K)/(XM*YM)
             KLAST=K
             IF (AFRAC(QPTS).GE.AFRACMAX) GOTO 101
             IF (QPTS.EQ.NPTSMAX) THEN
              WRITE(6,9998) QPTS
 9998         FORMAT(1x,' CAUTION from INTEGRATE_HEAVY_ION_UPSETS:',
     &                /,'    Maximum Array size reached: ',
     &                /,'    QPTS = ',I4)
              GOTO 101
             ENDIF
         ENDIF
 100  CONTINUE
 101  CONTINUE

C     Make sure we catch the plateau:
      IF (KLAST.LT.KLAST0) THEN
          IF (QPTS.LT.NPTSMAX) THEN
            QPTS=QPTS+1
```

```
             AFRAC(QPTS)=XSECT(KLAST0)/(XM*YM)
             QC(QPTS)=LETG(KLAST0)*(ZM+FUNNELM)*1.033E-5
          ENDIF
      ELSE
          KLAST0=KLAST
      ENDIF

C     Also store last value with zero cross-section:
      QC(1)=LETG(KFIRST)*(ZM+FUNNELM)*1.033E-5
C     Following should be zero:
      AFRAC(1)=XSECT(KFIRST)/(XM*YM)

      RETURN
      END
```

```fortran
      SUBROUTINE INTEGRATE_PROTON_UPSETS(NPTS,EN,FLUX,XSECT,SEU_RATE)
C
C     Performs numerical integration of flux X cross-section integral
C     for proton-induced SEUs.
C
C     Inputs:  NPTS = number of points in input arrays
C              EN   = array of proton energies (in MeV)
C              FLUX = array of proton differential flux
C                     (in protons/m2-s-sr-MeV), evaluated at EN
C              XSECT= array of SEU cross-sections, in 1.0E-12 cm2/bit,
C                     evaluated at EN
C     Output:  SEU_RATE = #SEUs/s/bit
C
C
C      Written by:    Allan J. Tylka
C                     Code 7654
C                     Naval Research Laboratory
C                     Washington, DC 20375-5352
C                     tylka@crs2.nrl.navy.mil
C
C      Last update:   29 March 1996
C
C----------------------------------------------------------------------
C
       IMPLICIT NONE
       INTEGER*4 I,NPTS
       REAL*4 EN,FLUX,XSECT
       REAL*4 FOURPI,DE,XINT,SEU_RATE
       DIMENSION EN(1),FLUX(1),XSECT(1)

       FOURPI=16.0*ATAN(1.0)
       SEU_RATE=0.0
       DO 200 I=1,NPTS-1
       DE=EN(I+1)-EN(I)
       XINT=0.5*(FLUX(I)*XSECT(I) + FLUX(I+1)*XSECT(I+1))
       SEU_RATE=SEU_RATE+XINT*DE
 200   CONTINUE

C
C      Now apply some factors:
C      Convert flux from /m2 to /cm2: 1.0E-4
C      Cross-section in units of 1.0E-12 cm2
C      Effective geometry factor = 4pi
C      Planar detector has a geometry factor of 2pi (top & bottom);
C      but the loss of projected area due to the obliquity factor is
C      compensated for by the sec-theta increase in pathlength through
C      the sensitive volume.

       SEU_RATE=FOURPI*SEU_RATE*1.0E-16
       RETURN
       END
```

```fortran
      REAL*4 FUNCTION INTERPOLATE_XSECT_TABLE(NSV,XV,YV,E)
C
C       Function does linear interpolation in a table to evaluate
C       the SEU cross-section
C
C        Inputs:  NSV:   number of entries in the table
C                 XV:    x-coordinates of table
C                 YV:    y-coordinates of table
C                 E:     x-value at which cross-section is required

C        Output:  SEU cross-section
C
C        NOTE:  returned value will be zero at E < XV(1)
C               returned value will be YV(NSV) at E > YV(NSV)
C               otherwise, linearly-interpolated.  It is the user's
C               responsibility to make sure the table is ordered as
C               monotonically-increasing XV values.
C
C
C       Written by:    Allan J. Tylka
C                      Code 7654
C                      Naval Research Laboratory
C                      Washington, DC 20375-5352
C                      tylka@crs2.nrl.navy.mil
C
C       Last update:   29 March 1996
C
C-----------------------------------------------------------------------------
C
      IMPLICIT NONE
      INTEGER*4 NSV,I
      REAL*4 XV,YV,E
      DIMENSION XV(1),YV(1)

      INTERPOLATE_XSECT_TABLE = 0.

      IF (NSV.LE.0 .OR. E.LT.XV(1)) RETURN

      IF (E.GT.XV(NSV)) THEN
          INTERPOLATE_XSECT_TABLE=YV(NSV)

      ELSE

          DO 100 I=2,NSV
             IF (E.LT.XV(I)) THEN
             INTERPOLATE_XSECT_TABLE=
     &       YV(I-1)+(E-XV(I-1))*(YV(I)-YV(I-1))/(XV(I)-XV(I-1))
             GOTO 200
             ENDIF
 100      CONTINUE
 200      CONTINUE

      ENDIF

      IF (INTERPOLATE_XSECT_TABLE.LT.0.) INTERPOLATE_XSECT_TABLE=0.0

      RETURN
      END
```

```
      PROGRAM LETSPEC_DRIVER

      IMPLICIT NONE
      CHARACTER*80 INFILE,OUTFILE
      REAL*4 LETMINMG,LETMAXMG,LETMIN,LETMAX,ELOWER,EUPPER
      REAL*4 EMINCUT,EMAXCUT
      CHARACTER*12 TARGET
      INTEGER*4 MARR,NELM,LARR,IZMIN,IZMAX,IZLO,IZUP,M,L
      PARAMETER (MARR=5000,NELM=92,LARR=1002)
      REAL*4 INPUT_FLUX(NELM,MARR),SPECT(LARR),DIFSPEC(LARR)
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE,IDIFSPEC
C
C
C     Get parameters of LETSPEC calculation:
C
      CALL INILET(INFILE,LETMINMG,LETMAXMG,IZMIN,IZMAX,EMINCUT,
     *            EMAXCUT,TARGET,OUTFILE,IDIFSPEC)

C     Unload input particle flux file into array:
C
      CALL UNLOAD_PARTIAL_FLUX(INFILE,IZMIN,IZMAX,EMINCUT,EMAXCUT,
     *                         ELOWER,EUPPER,M,IZLO,IZUP,
     *                         INPUT_FLUX)
C
C
C     Now do integral LET spectrum calculation:
C
      CALL CREME96_LETSPEC(LETMINMG,LETMAXMG,TARGET,
     *             ELOWER,EUPPER,M,IZLO,IZUP,
     &             INPUT_FLUX,
     &             VERSION_NUMBER,PROGRAM_CODE,IDIFSPEC,
     &             LETMIN,LETMAX,L,SPECT,DIFSPEC)
C
C
C     Now write integral LET spectrum to output file:
C
      CALL OUTPUT_CREME96_LETSPEC(LETMIN,LETMAX,L,
     *                            IZLO,IZUP,EMINCUT,EMAXCUT,TARGET,
     *                            VERSION_NUMBER,PROGRAM_CODE,
     *                            INFILE,
     *                            SPECT,
     *                            OUTFILE)
C
C     Also output differential LET spectrum
C
      IF (IDIFSPEC.EQ.1)
     *CALL OUTPUT_CREME96_DIFLET(LETMIN,LETMAX,L,
     *                            IZLO,IZUP,EMINCUT,EMAXCUT,TARGET,
     *                            VERSION_NUMBER,PROGRAM_CODE,
     *                            INFILE,
     *                            DIFSPEC,
     *                            OUTFILE)

C
      STOP
      END
```

```
      SUBROUTINE LOAD_GTF(GTRANSFILE)
C
C     Loads geomagnetic transmission function from specified file
C     into a common block for later use.
C
      IMPLICIT NONE
      CHARACTER*80 GTRANSFILE
      INTEGER*4 NGTF,IGTF,I,IGTFUNIT,IVER,NHEADER,STAT,CREME96_OPEN
      REAL*4 R,GTF
      PARAMETER (NGTF=1001)
      COMMON/GTFDAT/IGTF,R(NGTF),GTF(NGTF)
      LOGICAL IEXIST,CREME96_INQUIRE
      DATA IGTFUNIT/15/
C
C     First see if GTRANSFILE exists:
c      INQUIRE(FILE='USER:'//GTRANSFILE,EXIST=IEXIST)
      iexist = creme96_inquire(gtransfile,'user')
      IF (.NOT. IEXIST) THEN
         WRITE(6,999) GTRANSFILE
 999     FORMAT(' Geomagnetic Transmission File = ',/,
     &               1x,A80,/' not found.  Job aborted.')
         STOP
      ENDIF
      CALL CHECK_CREME96_VERSION(GTRANSFILE,IVER)

c      OPEN(UNIT=IGTFUNIT,READONLY,SHARED,STATUS='OLD',
c    &       FILE='USER:'//GTRANSFILE)
      stat = creme96_open(gtransfile,'user',igtfunit,'old')
C
C     Get pass header lines:

      IF (IVER.EQ.101) THEN
         READ(IGTFUNIT,*)
         READ(IGTFUNIT,*)
      ELSEIF (IVER.GE.102) THEN
         READ(IGTFUNIT,*) NHEADER
         DO I=1,NHEADER
            READ(IGTFUNIT,*)
         ENDDO
      ENDIF


C        Now begin real read-in.
         DO 5 I=1,NGTF
            READ(IGTFUNIT,*,END=6) R(I),GTF(I)
 5       CONTINUE
 6       CONTINUE
         IGTF=I-1
         CLOSE(15)

      RETURN
      END
```

```
      SUBROUTINE LOAD_SEP_QSTATES
C
C     Loads Solar Energetic Particle Ionic Charge State Distribution
C     from datafile into COMMON/SEP_QSTATES

C     11-17-97: IMPLICIT NONE and variable-type declarations added.

      IMPLICIT NONE
      REAL*4 SEP_QSTATES
      COMMON/SEP_QSTATES/SEP_QSTATES(30,30)
      INTEGER*4 ISQUNIT
      DATA ISQUNIT/18/
      LOGICAL IEXIST,CREME96_INQUIRE
      INTEGER*4 STAT,CREME96_OPEN


      INTEGER*4 J,NLINES,ILINE,KMIN,KMAX,IDUM,K

C     First see if QSTATES file is available here:
c      INQUIRE(FILE='CREME96:QSTATE.DAT',EXIST=IEXIST)
      iexist = creme96_inquire('qstate.dat','cr96tables')
      IF (.NOT.IEXIST) THEN
          WRITE(6,999)
 999      FORMAT(' File CREME96:QSTATE.DAT not found.  Abort.')
          STOP
      ELSE
c      OPEN(UNIT=ISQUNIT,READONLY,SHARED,STATUS='OLD',
c    &      FILE='CREME96:QSTATE.DAT')
      stat = creme96_open('qstate.dat','cr96tables',isqunit,'old')

      DO 500 J=1,30
         NLINES=(J-1)/8+1
         DO 400 ILINE=1,NLINES
            KMIN=(ILINE-1)*8+1
            KMAX=ILINE*8
            IF (KMAX.GT.J)KMAX=J
            READ(ISQUNIT,9006) IDUM,(SEP_QSTATES(J,K),K=KMIN,KMAX)
9006        FORMAT(I3,8F8.5)
 400     CONTINUE
 500  CONTINUE
      CLOSE(ISQUNIT)
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE LOAD_TRAPPED_PROTONS(TRAPDFILE)
C
C     Subroutine to unload CREME96 trapped proton spectrum from specified
C     file into a common bloc, for later combination with non-trapped fluxes.
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C     Last update:   18 November 1997
C
C
C-----------------------------------------------------------------------
C
C
      IMPLICIT NONE
      CHARACTER*80 TRAPDFILE,ILINE

      INTEGER*4 MAXSPEC,N,NZ,NZT,i,ITRPSPEC
      INTEGER*4 IVER,NHEADER,STAT,CREME96_OPEN,ILONG
      REAL*4 ENTRP,FLUXTRP,EL,EU
      PARAMETER (MAXSPEC=5000)
      COMMON/TRPDAT/ITRPSPEC,ENTRP(MAXSPEC),FLUXTRP(MAXSPEC)

      CALL CHECK_CREME96_VERSION(TRAPDFILE,IVER)

      stat = creme96_open(TRAPDFILE,'user',10,'old')
      ILONG=INDEX(TRAPDFILE,'.')

      IF (TRAPDFILE(ILONG+1:ILONG+2).EQ.'TR' .or.
     &    TRAPDFILE(ILONG+1:ILONG+3).EQ.'tr') THEN

         IF (IVER.GE.102) THEN
         READ(10,*) NHEADER
         DO i=1,NHEADER
            READ(10,110) ILINE
 110        FORMAT(A80)
         ENDDO
         ENDIF

         read(10,*) el,eu,n,nz,nzt
         IF (NZ.NE.1) THEN
         WRITE(6,999)
 999     FORMAT(1x,' WARNING: No proton spectrum in input file:',
     &        /,1x,' STOP in LOAD_TRAPPED_PROTONS')
         STOP
         ENDIF
C
C        Calculate abscissae (energy values)

         ENTRP(1)=el
         ENTRP(N)=eu
         do 100 i=2,N-1
         ENTRP(i)=el*(eu/el)**(float(i-1)/float(n-1))
100      continue
C
C        Read blank line
```

```
            read(10,110) ILINE

c           read in the flux

            IF (N.GT.MAXSPEC) N=MAXSPEC

            read(10,*)(fluxtrp(i),i=1,n)

            CLOSE(10)

C       Eliminate end-of-file zeroes from returned spectrum:

        ITRPSPEC=0
        DO 1000 I=1,N
           IF (FLUXTRP(I).GT.0.0) ITRPSPEC=I
 1000   CONTINUE

        ELSE

          WRITE(6,9999) TRAPDFILE
 9999     FORMAT(1x,' Specified TRAPPED PROTON FILE = ',
     &          /,1x,A80,
     &          /,1x,' does not appear to be a CREME96-generated file.',
     &          /,1x,' STOP')

        ENDIF

        RETURN
        end
```

```
      REAL*4 FUNCTION MAGNETIC_RIGIDITY(EK,Q,A)
C
C     Function to calculate the magnetic rigidity (in GV/c)
C     given inputs:
C               EK = kinetic energy per nucleon in MeV/amu
C               Q = charge
C               A = mass number (> 0 for ions; 0 for electrons)
C

      IMPLICIT REAL*8 (D)
      REAL*4 EK,Q,A
      DATA DAMU/0.9315016D0/
      DATA DELEC/0.00051099906/

      DK=EK/1000.D0
      DA=A
      DQ=Q
C     DR= gamma*beta

      IF (DA.GT.0.0) THEN
C     Ion case:
      DR=(1.D0+DK/DAMU)**2-1.D0
      DR=DSQRT(DR)*DA/DQ*DAMU

      ELSEIF (DA.EQ.0) THEN
C     Electron case:
      DR=(1.D0+DK/DELEC)**2-1.D0
      DR=DSQRT(DR)*DELEC
      ENDIF


      MAGNETIC_RIGIDITY=DR

      RETURN
      END
```

```
      program MAKE_DEDX_TABLE
C
C     Makes two-column energy vs. dE/dx tables from CREME96 software.
c
      IMPLICIT NONE
      INTEGER*4 NBINMAX,STAT,CREME96_OPEN
      PARAMETER (NBINMAX=5000)
      INTEGER*4 NBINS,K,IZ,OUTUNIT,IMAT
      REAL*4 E(NBINMAX)
      REAL*4 EMIN,EMAX,EN,DE,Z,AN,DEDX,STPOW
      CHARACTER*80 OUTFILE
      CHARACTER*12 MATERIAL,MATS(2)
      DATA MATS/'ALUMINUM    ','SILICON     '/
      DATA OUTUNIT/8/
      INTEGER*4 IERR,IACCEPT
      DATA IERR/0/

      MATERIAL='ALUMINUM    '
      WRITE(6,8000)
 8000 FORMAT(' This program will make a table of stopping power',
     &       ' in aluminum for the specified'
     &     /,' nuclei.  The table will be specified for NBINS',
     &       ' logarithmically-spaced energy',
     &     /,' values between limits EMIN and EMAX (in MeV/nuc)')
 8101 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,8001) NBINMAX
 8001 FORMAT(' Enter EMIN, EMAX (in MeV/nuc) and NBINS (<',I5,')')
      READ(*,*,ERR=8101,IOSTAT=IERR) EMIN,EMAX,NBINS
      IF (NBINS.GT.NBINMAX) NBINS=NBINMAX
      WRITE(6,8002) EMIN,EMAX,NBINS
 8002 FORMAT(1x,' EMIN = ',E13.5,'  EMAX = ',E13.5,' NBINS = ',I5)

 8103 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,8003)
 8003 FORMAT(' Select material: Enter 1 for Aluminum; 2 for Silicon')
      READ(*,*,ERR=8103,IOSTAT=IERR) IMAT
      MATERIAL=MATS(IMAT)
      WRITE(6,8004) MATERIAL
 8004 FORMAT(1x,' Material = ',A12)
C
C    Compute energies on logaritmically-spaced grid

      DE=(EMAX/EMIN)**(1./(NBINS-1.))
      E(1)=EMIN
      DO K=2,NBINS-1
         E(K)=E(K-1)*DE
      END DO
      E(NBINS)=EMAX

 9102 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9002)
 9002 FORMAT(1x,' Enter name of output file:')
      READ(*,2,ERR=9102,IOSTAT=IERR) OUTFILE
 2    FORMAT(A80)
      CALL CHECK_OUTPUT_FILE(OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) THEN
      WRITE(6,1011) OUTFILE(1:75)
```

```
          WRITE(6,1012)
 1011     FORMAT(1x,' Previous try at OUTPUT name = ',/,5x,A75)
 1012     FORMAT(1x,' Try again.')
          GOTO 9102
          ENDIF

c         open(unit=OUTUNIT,status='new',file='USER:'//OUTFILE)
          stat = creme96_open(outfile,'user',outunit,'new')
          WRITE(OUTUNIT,191) EMIN,EMAX,NBINS
 191      FORMAT(1x,'%EMIN = ',E13.5,'   EMAX = ',E13.5,' NBINS = ',I5)


 114      CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,9003)
 9003     FORMAT(1x,' Enter first desired IZ,A values for table: '
     &         /,8x,' IZ = atomic number',
     &         /,8x,'  A = mass number')
          READ(*,*,ERR=114,IOSTAT=IERR) IZ,AN

 115      CONTINUE
          Z=IZ*1.0
          IF (IZ.LT.0 .or. IZ.GT.92) THEN
           WRITE(6,9004) IZ
 9004      FORMAT(1x,' Invalid atomic number: ',I6,'  Please try again.')
           GOTO 114
          ENDIF


          WRITE(OUTUNIT,190) IZ,AN,MATERIAL
 190      FORMAT(1x,'%Energy (MeV/nuc)'
     &         '  dE/dx (MeV/(g/cm2))',
     &         '  for Z = ',I2,' A = ',F6.2,' in ',A12)

C
          DO 80 K=1,NBINS
             EN=E(K)
             DEDX=STPOW(EN,Z,AN,MATERIAL)
C            STPOW calculates energy loss in MeV/nuc:
             DEDX=AN*DEDX
             write(OUTUNIT,34) EN,DEDX
34           FORMAT(2X,E12.5,2X,E12.5)
80        continue

 9106     CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(6,9006)
 9006     FORMAT(1x,' Enter next IZ,A value (0 0 to end program):')
          READ(*,*,ERR=9106,IOSTAT=IERR) IZ,AN
          IF (IZ.GT.0) GOTO 115
          CLOSE(OUTUNIT)
          STOP
          end
```

```
      SUBROUTINE MAKE_DIFLET_SPECTRUM(LETMIN,LETMAX,LIN,SPECT,DIFSPEC)
C
C     Makes SIMPLE (ie., needs work) numerical differentiation of
C     integral LET spectrum to produce a differential LET spectrum.
C
      IMPLICIT NONE
      REAL*4 LETMIN,LETMAX
      INTEGER*4 LIN,L,K,LARR
      REAL*4 SPECT,DIFSPEC
      DIMENSION SPECT(1),DIFSPEC(1)
      REAL*8 DY,DL,LETG
      PARAMETER(LARR=1002)
      DIMENSION LETG(LARR)

C     Fill array of LET values:
      L=LIN
      IF (L.GT.LARR) L=LARR

      DL=(LETMAX/LETMIN)**(1./FLOAT(L-1))
      LETG(1)=LETMIN
      DIFSPEC(1)=0.0
      DO 400 K=2,L-1
         LETG(K)=LETG(K-1)*DL
         DIFSPEC(K)=0.0
 400  CONTINUE
      LETG(L)=LETMAX
      DIFSPEC(L)=0.0

C     Now calculate differential LET spectrum.

      DL=LETG(2)-LETG(1)
      IF (SPECT(2).GT.0.)
     &   DIFSPEC(1)=-SPECT(1)*ALOG(SPECT(2)/SPECT(1))/DL
      DO 500 K=2,L-1
         IF (SPECT(K+1).GT.0.0) THEN
             DL=(LETG(K+1)-LETG(K-1))
             DY=-ALOG(SPECT(K+1)/SPECT(K-1))
         ELSE
             IF (SPECT(K).GT.0.0) THEN
             DL=LETG(K)-LETG(K-1)
             DY=-ALOG(SPECT(K)/SPECT(K-1))
             ENDIF
         ENDIF
         DIFSPEC(K)=SPECT(K)*DY/DL
 500  CONTINUE

      RETURN
      END
```

```
      program MAKE_FLUX_FIG
C
c     Rewrites flux output file from CREME96/UPROP format
C     into a .FIG file.
c
      IMPLICIT NONE
      INTEGER*4 MARR,NELM,STAT,CREME96_OPEN
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 FLUX(NELM,MARR),E(MARR)
      REAL*4 EL,EU,ETEMP,EMINCUT,EMAXCUT
      INTEGER*4 IACCEPT,IFILETYPE,IZMIN,IZMAX,M,IZLO,IZUP
      CHARACTER*80 INFILE,OUTFILE
      CHARACTER*80 ILINE,TEMPLINE
      INTEGER*4 IZTARG,NHMAX,LINEMAX,ICOUNT
      PARAMETER (NHMAX=30)
      DIMENSION ILINE(NHMAX)
      INTEGER*4 I,OUTUNIT,FHDUNIT,NHEADER,IZDUM,ILONG
      LOGICAL ZERO
      INTEGER*4 IERR
      DATA IERR/0/

      OUTUNIT=42
      FHDUNIT=43
 112  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9001)
 9001 FORMAT(1x,' Enter name of input file',
     &           ' (something.FLX, .TFX, or .TR*):')
      READ(*,2,ERR=112,IOSTAT=IERR) INFILE
 2    FORMAT(A80)
      IFILETYPE=3
      WRITE(6,1020) INFILE
 1020 FORMAT(' Input Flux File =',/,1x,A80)
      CALL CHECK_FILE(IFILETYPE,INFILE,IACCEPT)
      IF (IACCEPT.NE.0) GOTO 112
C
C     Now unload fluxes from this file:
      EMINCUT=0.0
      EMAXCUT=1.0E+24
      CALL UNLOAD_PARTIAL_FLUX(INFILE,IZMIN,IZMAX,EMINCUT,EMAXCUT,
     &                              EL,EU,M,IZLO,IZUP,
     *                              FLUX)

C     Calculate abscissae (energies)

      e(1)=EL
      e(M)=EU
      do 10 i=2,M-1
         e(i)=el*(eu/el)**(float(i-1)/float(M-1))
 10   continue

C     Now start to copy information to new file:

 9102 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9002)
 9002 FORMAT(1x,' Enter name of output file (something.FIG):')

      READ(*,2,ERR=9102,IOSTAT=IERR) OUTFILE
      CALL CHECK_OUTPUT_FILE(OUTFILE,IACCEPT)
```

```
        IF (IACCEPT.NE.0) THEN
        WRITE(6,1010) INFILE(1:75)
        WRITE(6,1011) OUTFILE(1:75)
        WRITE(6,1012)
 1010   FORMAT(1x,' INPUT file = ',/,5x,A75)
 1011   FORMAT(1x,' Previous try at OUTPUT name = ',/,5x,A75)
 1012   FORMAT(1x,' Try again.')
        GOTO 9102
        ENDIF

c       open(unit=OUTUNIT,status='new',file='USER:'//OUTFILE)
        stat = creme96_open(outfile,'user',outunit,'new')
        WRITE(OUTUNIT,185) INFILE(1:70)
 185    FORMAT('*',A50)
C
C       Now add FIGGEN header information:
C       Write FIGGEN header
c        OPEN(UNIT=FHDUNIT,status='old',readonly,shared,
c     &       file='CREME96:FLX_FIG.header')
        stat = creme96_open('flx_fig.header','cr96tables',fhdunit,'old')
6       CONTINUE
        READ (FHDUNIT,2,END=8) ILINE(1)
        WRITE(OUTUNIT,2) ILINE(1)
        GOTO 6
8       CONTINUE
        CLOSE(FHDUNIT)

        ILONG=LEN(INFILE)
        CALL CAPITALIZE_STRING(INFILE,ILONG)

        WRITE(OUTUNIT,186) INFILE(1:50)
 186    FORMAT('ST 0.3 100 1.0E+5 0.//^^',A50)
C       Transfer header information from input file
        CALL UNLOAD_HEADERS(INFILE,NHMAX,ILINE,LINEMAX)
        IF (LINEMAX.GT.0) THEN
        DO 100 I=1,LINEMAX
           TEMPLINE=ILINE(I)
           TEMPLINE='*'//TEMPLINE(2:80)
           WRITE(OUTUNIT,2) TEMPLINE
 100    CONTINUE
        ENDIF

 114    CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(6,9003)
 9003   FORMAT(1x,' Enter desired IZMIN,IZMAX for FIGure: ')
        READ(*,*,ERR=114,IOSTAT=IERR) IZMIN,IZMAX
 115    CONTINUE

        DO 200 IZTARG=IZMIN,IZMAX
        IF (IZTARG.LT.0 .or. IZTARG.GT.92) THEN
            WRITE(6,9004) IZTARG
 9004       FORMAT(1x,' Invalid atomic number: ',I6)
            GOTO 200
        ENDIF
C
C       Does this file contain the Z value of interest?

        IF (IZTARG.LT.IZLO .or. IZTARG.GT.IZUP) THEN
           WRITE(6,9005) IZLO,IZUP,IZTARG
```

```
 9005      FORMAT(1x,' In this file: IZLO = ',I5,'  IZHI =',I5,
     &          /,1x,' IZ = ',I5,' not found here.')
           GOTO 200
        ENDIF

        WRITE(OUTUNIT,30) IZTARG
 30     FORMAT('* IZ = ',I3)
C
        ZERO=.TRUE.
        DO 50 I=1,M
           IF (FLUX(IZTARG,I).GT.0.) THEN
               ZERO=.FALSE.
               GOTO 55
           ENDIF
 50     CONTINUE
 55     CONTINUE

        IF(.NOT.ZERO) THEN
        IF (MOD(IZTARG,2) .EQ.0) THEN
                WRITE(42,31) M
        ELSE
                WRITE(42,32) M
        ENDIF
 31     FORMAT('FRENCH ',I4,' 0.01 0')
 32     FORMAT('FRENCH ',I4,' 0.10 2')

        DO 80 I=1,M
        IF (FLUX(IZTARG,I).GT.0.0) THEN
            write(OUTUNIT,34) e(i),flux(IZTARG,i)
        ENDIF
34      FORMAT(2X,E12.5,2X,E12.5)
80      continue

        ICOUNT=ICOUNT+1
        IF (ICOUNT.GT.3) ICOUNT=1
        ETEMP=E(M)*(1.+ICOUNT*0.05)
        WRITE(OUTUNIT,90) ETEMP,FLUX(IZTARG,M),IZTARG
90      FORMAT('STHC -0.2 ',E12.5,2x,E12.5,' 0//',I2)

        ENDIF
200     CONTINUE

        CLOSE(OUTUNIT)
        STOP
        end
```

```
      program MAKE_FLUX_TABLE
C
c     Rewrites flux output file from CREME96/UPROP format
C     into a two column table.
c
      IMPLICIT NONE
      INTEGER*4 MARR,NELM,STAT,CREME96_OPEN
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 FLUX(NELM,MARR),E(MARR)
      REAL*4 EL,EU,EMINCUT,EMAXCUT
      INTEGER*4 IACCEPT,IFILETYPE,IZTARG,M,IZLO,IZUP
      CHARACTER*80 INFILE,OUTFILE
      INTEGER*4 I,OUTUNIT,NHEADER,IZDUM
      INTEGER*4 IERR
      DATA IERR/0/

      OUTUNIT=42
 112  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9001)
 9001 FORMAT(1x,' Enter name of input file',
     &       ' (something.FLX, .TFX, or .TR*):')
      READ(*,2,ERR=112,IOSTAT=IERR) INFILE
 2    FORMAT(A80)
      IFILETYPE=3
      WRITE(6,1020) INFILE
 1020 FORMAT(' Input Flux File =',/,1x,A80)
      CALL CHECK_FILE(IFILETYPE,INFILE,IACCEPT)
      IF (IACCEPT.NE.0) GOTO 112
C
C     Now unload fluxes from this file:
      IZDUM=0
      EMINCUT=0.0
      EMAXCUT=1.0E+24
      CALL UNLOAD_PARTIAL_FLUX(INFILE,IZDUM,IZDUM,EMINCUT,EMAXCUT,
     &                                  EL,EU,M,IZLO,IZUP,
     *                                  FLUX)

C     Calculate abscissae (energies)

      e(1)=EL
      e(M)=EU
      do 10 i=2,M-1
         e(i)=el*(eu/el)**(float(i-1)/float(M-1))
10    continue

C     Now start to copy information to new file:

 9102 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9002)
 9002 FORMAT(1x,' Enter name of output file (something.DAT):')
      READ(*,2,ERR=9102,IOSTAT=IERR) OUTFILE
      CALL CHECK_OUTPUT_FILE(OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) THEN
      WRITE(6,1010) INFILE(1:75)
      WRITE(6,1011) OUTFILE(1:75)
      WRITE(6,1012)
 1010 FORMAT(1x,' INPUT file = ',/,5x,A75)
 1011 FORMAT(1x,' Previous try at OUTPUT name = ',/,5x,A75)
```

```
 1012   FORMAT(1x,' Try again.')
        GOTO 9102
        ENDIF

c       open(unit=OUTUNIT,status='new',file='USER:'//OUTFILE)
        stat = creme96_open(outfile,'user',outunit,'new')
        WRITE(OUTUNIT,185) INFILE(1:79)
 185    FORMAT(1x,A79)
        CALL CHECK_HEADER_LENGTH(INFILE,NHEADER)
        CALL COPY_HEADERS(INFILE,NHEADER,OUTUNIT)

 114    CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(6,9003)
 9003   FORMAT(1x,' Enter desired IZ value for table: ')
        READ(*,*,ERR=114,IOSTAT=IERR) IZTARG
 115    CONTINUE
        IF (IZTARG.LT.0 .or. IZTARG.GT.92) THEN
         WRITE(6,9004) IZTARG
 9004    FORMAT(1x,' Invalid atomic number: ',I6,'  Please try again.')
         GOTO 114
        ENDIF
C
C       Does this file contain the Z value of interest?

        IF (IZTARG.LT.IZLO .or. IZTARG.GT.IZUP) THEN
           WRITE(6,9005) IZLO,IZUP,IZTARG
 9005      FORMAT(1x,' In this file: IZLO = ',I5,'  IZHI =',I5,
     &          /,1x,' IZ = ',I5,' not found here.',
     &          /,1x,' Please try again.')
           GOTO 114
        ENDIF

        WRITE(OUTUNIT,190) IZTARG
190     FORMAT(1x,'%Energy (in MeV/nuc)'
     &       '  vs. Flux (in particles/m2-s-sr-MeV/nuc)',
     &       '  for Z = ',I2)
C
        DO 80 I=1,M
        IF (FLUX(IZTARG,I).GT.0.0) THEN
            write(OUTUNIT,34) e(i),flux(IZTARG,i)
        ENDIF
34      FORMAT(2X,E12.5,2X,E12.5)
80      continue

 9106   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(6,9006)
 9006   FORMAT(1x,' Enter next Z value (0 to end program):')
        READ(*,*,ERR=9106,IOSTAT=IERR) IZTARG
        IF (IZTARG.GT.0) GOTO 115

        CLOSE(OUTUNIT)
        STOP
        end
```

```
        program MAKE_LETSPEC_FIG
c
c       Rewrites integral LET spectrum output file from CREME96/UPROP format
C       to a FIGGEN file
c
        IMPLICIT NONE
        INTEGER*4 MARR,STAT,CREME96_OPEN
        PARAMETER (MARR=5000)
        REAL*4 FLUX(MARR),E(MARR)
        INTEGER*4 IACCEPT,IFILETYPE,M
        CHARACTER*80 INFILE,OUTFILE,HEADER
        CHARACTER*80 ILINE,TEMPLINE
        INTEGER*4 NHMAX,LINEMAX
        PARAMETER (NHMAX=30)
        DIMENSION ILINE(NHMAX)
        INTEGER*4 I,OUTUNIT,FHDUNIT,IERR,ISPECTYPE,ILONG
        DATA IERR/0/
        LOGICAL INTLET,DIFLET

        OUTUNIT=42
        FHDUNIT=43
        ILONG=0

 112    CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(6,9001)
 9001   FORMAT(1x,' Enter name of input file: ',
     &       /,5x,'(something.LET for an integral LET spectrum;',
     &       /,5x,' something.DLT for a differential LET spectrum)')
        READ(*,2,ERR=112,IOSTAT=IERR) INFILE
 2      FORMAT(A80)
        INTLET=.FALSE.
        DIFLET=.FALSE.
        ILONG=INDEX(INFILE,'.')
        IF (ILONG.EQ.0) GOTO 112
        IF (INFILE(ILONG+1:ILONG+3).EQ.'LET' .or.
     &      INFILE(ILONG+1:ILONG+3).EQ.'let') INTLET=.TRUE.
        IF  (INFILE(ILONG+1:ILONG+3).EQ.'DLT' .or.
     &       INFILE(ILONG+1:ILONG+3).EQ.'dlt') DIFLET=.TRUE.

        IF (.NOT.INTLET .and. .NOT.DIFLET) THEN
 111    CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(6,9000)
 9000   FORMAT(1x,' LET file type not known.  Please specify type:',
     &       /,1x,' 0=integral or 1=differential')
        READ(2,*,ERR=111,IOSTAT=IERR) ISPECTYPE
        IF (ISPECTYPE.LT.0 .or. ISPECTYPE.GT.1) GOTO 111
        IF (ISPECTYPE.EQ.0) INTLET=.TRUE.
        IF (ISPECTYPE.EQ.1) DIFLET=.TRUE.
        ENDIF

        IF (INTLET) IFILETYPE=5
        IF (DIFLET) IFILETYPE=6
        WRITE(6,1020) INFILE
 1020   FORMAT(' Input Flux File =',/,1x,A80)
        CALL CHECK_FILE(IFILETYPE,INFILE,IACCEPT)
        IF (IACCEPT.NE.0) GOTO 112

C       Now unload fluxes from this file:
```

```
      CALL UNLOAD_LET_SPECTRUM(INFILE,E,FLUX,M)

C     Now start to copy information to new file:

 9102 CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9002)
 9002 FORMAT(1x,' Enter name of output file (something.FIG):')
      READ(*,2,ERR=9102,IOSTAT=IERR) OUTFILE

      CALL CHECK_OUTPUT_FILE(OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) THEN
      WRITE(6,1010) INFILE(1:75)
      WRITE(6,1011) OUTFILE(1:75)
      WRITE(6,1012)
 1010 FORMAT(1x,' INPUT file = ',/,5x,A75)
 1011 FORMAT(1x,' Previous try at OUTPUT name = ',/,5x,A75)
 1012 FORMAT(1x,' Try again.')
      GOTO 9102
      ENDIF

c     open(unit=OUTUNIT,status='new',file='USER:'//OUTFILE)
      stat = creme96_open(outfile,'user',outunit,'new')
      WRITE(OUTUNIT,185) INFILE(1:70)
 185  FORMAT('*',A50)


C     Now add FIGGEN header information:
C     Write FIGGEN header
c      IF (INTLET) HEADER='CREME96:LETSPEC_FIG.header'
c      IF (DIFLET) HEADER='CREME96:DLTSPEC_FIG.header'
      IF (INTLET) HEADER='LETSPEC_FIG.header'
      IF (DIFLET) HEADER='DLTSPEC_FIG.header'
c      OPEN(UNIT=FHDUNIT,status='old',readonly,shared,file=HEADER)
      stat = creme96_open(header,'cr96tables',fhdunit,'old')
6     CONTINUE
      READ (FHDUNIT,2,END=8) ILINE(1)
      WRITE(OUTUNIT,2) ILINE(1)
      GOTO 6
8     CONTINUE
      CLOSE(FHDUNIT)

      ILONG=LEN(INFILE)
      CALL CAPITALIZE_STRING(INFILE,ILONG)

      IF (INTLET) WRITE(OUTUNIT,186) INFILE(1:50)
      IF (DIFLET) WRITE(OUTUNIT,187) INFILE(1:50)
 186  FORMAT('ST 0.3 1.0E+2 1.0E+3 0.//^^',A50)
 187  FORMAT('ST 0.3 1.0E+3 1.0E+3 0.//^^',A50)
C     Transfer header information from input file
      CALL UNLOAD_HEADERS(INFILE,NHMAX,ILINE,LINEMAX)
      IF (LINEMAX.GT.0) THEN
      DO 100 I=1,LINEMAX
         TEMPLINE=ILINE(I)
         TEMPLINE='*'//TEMPLINE(2:80)
         WRITE(OUTUNIT,2) TEMPLINE
 100  CONTINUE
      ENDIF

      IF (INTLET) WRITE(OUTUNIT,190)
```

```
      IF (DIFLET) WRITE(OUTUNIT,191)
190   FORMAT(1x,'*%LET (in MeV-cm2/g)'
     &       '  vs. Integral Flux (in particles/m2-s-sr)')
191   FORMAT(1x,'*%LET (in MeV-cm2/g)'
     & '  vs. Differential Flux (in particles/m2-s-sr-(MeV-cm2/g))')
C
      WRITE(42,31) M
 31   FORMAT('FRENCH ',I4,' 0.01 0')

      DO 80 I=1,M
      IF (FLUX(I).GT.0.0) THEN
          write(OUTUNIT,34) e(i),flux(i)
      ENDIF
34    FORMAT(2X,E12.5,2X,E12.5)
80    continue

      CLOSE(OUTUNIT)
      STOP
      end
```

```
      program MAKE_LETSPEC_TABLE
C
c     Rewrites integral LET output file from CREME96/UPROP format
C     into a two column table.
c
      IMPLICIT NONE
      INTEGER*4 MARR,STAT,CREME96_OPEN
      PARAMETER (MARR=5000)
      REAL*4 FLUX(MARR),E(MARR),LETMIN
      INTEGER*4 IACCEPT,IFILETYPE,M
      CHARACTER*80 INFILE,OUTFILE
      INTEGER*4 I,OUTUNIT,NHEADER,IERR,ISPECTYPE,ILONG
      DATA IERR/0/
      LOGICAL INTLET,DIFLET

      OUTUNIT=42
      ILONG=0
 112  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9001)
 9001 FORMAT(1x,' Enter name of input file: ',
     &      /,5x,'(something.LET for an integral LET spectrum;',
     &      /,5x,' something.DLT for a differential LET spectrum)')
      READ(*,2,ERR=112,IOSTAT=IERR) INFILE
 2    FORMAT(A80)
      INTLET=.FALSE.
      DIFLET=.FALSE.
      ILONG=INDEX(INFILE,'.')
      IF (ILONG.EQ.0) GOTO 112
      IF (INFILE(ILONG+1:ILONG+3).EQ.'LET' .or.
     &    INFILE(ILONG+1:ILONG+3).EQ.'let') INTLET=.TRUE.
      IF  (INFILE(ILONG+1:ILONG+3).EQ.'DLT' .or.
     &     INFILE(ILONG+1:ILONG+3).EQ.'dlt') DIFLET=.TRUE.

      IF (.NOT.INTLET .and. .NOT.DIFLET) THEN
 111  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9000)
 9000 FORMAT(1x,' LET file type not known.  Please specify type:',
     &      /,1x,' 0=integral or 1=differential')
      READ(2,*,ERR=111,IOSTAT=IERR) ISPECTYPE
      IF (ISPECTYPE.LT.0 .or. ISPECTYPE.GT.1) GOTO 111
      IF (ISPECTYPE.EQ.0) INTLET=.TRUE.
      IF (ISPECTYPE.EQ.1) DIFLET=.TRUE.
      ENDIF

      IF (INTLET) IFILETYPE=5
      IF (DIFLET) IFILETYPE=6
      WRITE(6,1020) INFILE
 1020 FORMAT(' Input Flux File =',/,1x,A80)
      CALL CHECK_FILE(IFILETYPE,INFILE,IACCEPT)
      IF (IACCEPT.NE.0) GOTO 112
C
C     Now unload fluxes from this file:
      CALL UNLOAD_LET_SPECTRUM(INFILE,E,FLUX,M)

C     Now start to copy information to new file:

 9201 CONTINUE
      CALL RETRY_INPUT(IERR)
```

```
      WRITE(6,9101)
9101  FORMAT(1x,' Enter starting LET value (in MeV-cm2/g)',
    &           ' to be included in output table.',
    &          /,'  (For most SEU applications, LET = 100 MeV-cm2/g',
    &           ' is appropriate.)')

      READ(*,*,ERR=9201,IOSTAT=IERR) LETMIN

9202  CONTINUE
      CALL RETRY_INPUT(IERR)
      WRITE(6,9002)
9002  FORMAT(1x,' Enter name of output file (something.DAT):')
      READ(*,2,ERR=9202,IOSTAT=IERR) OUTFILE
      CALL CHECK_OUTPUT_FILE(OUTFILE,IACCEPT)
      IF (IACCEPT.NE.0) THEN
      WRITE(6,1010) INFILE(1:75)
      WRITE(6,1011) OUTFILE(1:75)
      WRITE(6,1012)
1010  FORMAT(1x,' INPUT file = ',/,5x,A75)
1011  FORMAT(1x,' Previous try at OUTPUT name = ',/,5x,A75)
1012  FORMAT(1x,' Try again.')
      GOTO 9202
      ENDIF

c     open(unit=OUTUNIT,status='new',file='USER:'//OUTFILE)
      stat = creme96_open(outfile,'user',outunit,'new')
      WRITE(OUTUNIT,185) INFILE(1:50),LETMIN
185   FORMAT(1x,A50,' LET > ',E12.5)
      CALL CHECK_HEADER_LENGTH(INFILE,NHEADER)
      CALL COPY_HEADERS(INFILE,NHEADER,OUTUNIT)

      IF (INTLET) WRITE(OUTUNIT,190) LETMIN
      IF (DIFLET) WRITE(OUTUNIT,191) LETMIN
190   FORMAT(1x,'%LET (in MeV-cm2/g)'
    &      ' vs. Integral Flux (in particles/m2-s-sr);',
    &      ' LET >',E10.3)
191   FORMAT(1x,'%LET (in MeV-cm2/g)'
    & ' vs. Diff. Flux (in #/m2-s-sr-(MeV-cm2/g));',
    & ' LET >',E10.3)

C
      DO 80 I=1,M
      IF (FLUX(I).GT.0.0.and.E(I).GE.LETMIN) THEN
          write(OUTUNIT,34) e(i),flux(i)
      ENDIF
34    FORMAT(2X,E12.5,2X,E12.5)
80    continue

      CLOSE(OUTUNIT)
      STOP
      end
```

```
      SUBROUTINE MFP(ENERGY,IZ,IA,NAME,MEAN_FREE_PATH)
!---------------------------------------------------------------------
!     This subroutine computes the mean free path of any nuclide
!     (IZ,IA) in material NAME at energy ENERGY (MeV/N).
!
!     It must be linked with CREME96:ZTARGET.DAT
!---------------------------------------------------------------------
c     INCLUDE 'CREME96:ZCOMMON.CMN'
      CHARACTER*12 NAME
      REAL NA(28),IADJ(28),NASPM(28),DENS,ETAD
      INTEGER NZ(28),IGAS,NAS
      COMMON /TBLOCK/DENS,ETAD,IGAS,NAS,
     &               NZ,NA,IADJ,NASPM,
     &               NTOTAL,AVGZ,AVGZ2,AVGA,AVGI

      REAL IA,MEAN_FREE_PATH
      DATA AVOGADRO/6.022045E23/  !  particles/mole

      CALL ZTARGET(NAME)

!     Compute mulitiplicative factor

      FACT=1.E27*AVGA/AVOGADRO
      AVGS=0.
      DO J=1,NAS
        NNZ=NZ(J)
        AAN=NA(J)
        CALL SMASH(IZ,IA,NNZ,AAN,ENERGY,S)
        AVGS=AVGS+NASPM(J)*S
      END DO
      AVGS=AVGS/NTOTAL
      MEAN_FREE_PATH=AVGS/FACT

      RETURN
      END

      SUBROUTINE SMASH(IZ,IA,JZ,JA,E,CROSS_SECTION)
C---------------------------------------------------------------------
C     Computes nucleus-nucleus total reaction cross section
C
C     inputs:
C
C       E    = projectile energy (MeV/nucleon)
C       IZ   = projectile charge
C       IA   = projectile mass
C       JZ   = target charge
C       JA   = target mass
C
C     SUBROUTINE
C     SUBROUTINE LETAW(IZ,A,ENERGY,CROSS_SECTION)
C     SUBROUTINE OVERLAP(IA,JA,CROSS_SECTION)
C---------------------------------------------------------------------
      REAL IA,JA

C     Define constants

      PI=3.1415927

!     Zero cross section at zero energy
```

```
      IF (E.LE.0) THEN
        CROSS_SECTION=0.
        RETURN
      ENDIF

C     proton-proton reactions

      IF (IZ*IA*JZ*JA.LE.1.02) THEN
        IF (IZ.EQ.JZ) THEN
          CROSS_SECTION=PTOTAL(E,1)
        ELSE
          CROSS_SECTION=PTOTAL(E,2)
        ENDIF
        RETURN
      ENDIF

C     proton or neutron projectile

      IF (IZ*IA.LE.1.02) THEN
        CALL LETAW(JZ,JA,E,CROSS_SECTION)
        RETURN
      ENDIF

C     proton or neutron target

      IF (JZ*JA.LE.1.02) THEN
        CALL LETAW(IZ,IA,E,CROSS_SECTION)
        RETURN
      ENDIF

C     general nucleus-nucleus collision

      CALL OVERLAP(IA,JA,CROSS_SECTION)

      RETURN
      END

c     FUNCTION OVERLAP(IA,JA,CROSS_SECTION)
      subroutine OVERLAP(IA,JA,CROSS_SECTION)
!     Westfall mass-changing  10*pi*r**2=67.887
      REAL IA,JA
      CROSS_SECTION=67.887*(IA**(1./3.)+JA**(1./3.)-1.12)**2
      RETURN
      END

      FUNCTION PTOTAL(E,N)
C-----------------------------------------------------------------
C     This function computes the proton-proton and proton-neutron
C     total cross section according to empirical formulas.
C     It is valid for energies between 40 MeV and 1000 GeV.
C     If N=1 (pp), if N=2 (pn).  Energy in MeV.
C-----------------------------------------------------------------
      DIMENSION P(12,2),C(3)
      DATA P/293.3,1.99,35.0,15.02,2.925,548.3,
     & 1104.,-.1444,4091.,.1174,75100.,.05061,
     & 1623.,1.423,30.97,13.08,.9946,561.9,
     & 2677.,-.0586,25950.,.0691,1.E6,0./
      A=(P(1,N)/E)**P(2,N) + P(3,N)
      B=P(4,N)*TANH(P(5,N)*ALOG(E/P(6,N)))
      DO J=1,3
```

```
      K=2*J+5
      DC=E/P(K,N)
      WC=EXP(-DC)
      C(J)=WC + (1.-WC)*DC**P(K+1,N)
    END DO
    PTOTAL=(A+B)*C(1)*C(2)*C(3)
    RETURN
    END

    SUBROUTINE LETAW(IZ,IA,ENERGY,CROSS_SECTION)
!-------------------------------------------------------------------
!   This subroutine computes the total inelastic cross sections
!   of nuclides on protons.  The formula is taken from:
!   Letaw,J.R.,Silberberg,R.,and Tsao,C.H. 1983,Ap.J.Suppl.,
!   51,271.
!-------------------------------------------------------------------
    REAL IA
    E=1.-.62*EXP(-ENERGY/200.)*SIN(10.9/ENERGY**.28)
    T=45.*IA**.7
    T=(1.+.016*SIN(5.3-2.63*LOG(IA)))*T*E
    IF(IZ.EQ.2) T=.8*T
    IF(IZ.EQ.4) T=(1.+.75*EXP(-ENERGY/75.))*T
    CROSS_SECTION=T
    RETURN
    END
```

```
      SUBROUTINE CREME96_TRANSPORT(INPUT_FLUX,
     &                             ELOWER,EUPPER,M,IZLO,IZUP,
     &                             IPATH,UPATH0,TARGET,SHIELDFILE,
     &                             VERSION_NUMBER,PROGRAM_CODE,
     &                             OUTPUT_FLUX)
C
C**************************************************************************
C  This subroutine transports an input particle environment through a
C  specified thickness and type of shielding.  It takes account both
C  ionization energy loss (dE/dx) as well as energy-dependent nuclear
C  fragmentation.  The output is the particle enviroment (differential
C  fluxes vs. energy) inside the spacecraft, that is, 'behind' the specified
C  shielding.  This routine includes many refinements over the old CREME
C  transport routine ("INSIDE").  Specifically:
C
C  CREME96_TRANSPORT keeps track of projectile fragments; the old CREME
C  code ignored them.  This routine also uses improved Silberberg, Tsao,
C  and Barghouty energy-dependent fragmentation cross-sections.  Both of
C  these improvements can be important for thick shielding.
C
C  At present CREME96_TRANSPORT only does aluminum shielding; future
C  versions will also offer transport through other shielding materials.
C
C  CREME96_TRANSPORT is based on the "UPROP" code, as originally developed
C  by John R. Letaw of Severn Communication Corp. under contract to
C  the Gamma & Cosmic Ray Astrophysics Branch of Naval Research Laboratory
C  in 1989.  Significant improvements and "bug-extermination" have been
C  provided by A.F. Barghouty of Roanoke College.
C
C**************************************************************************
C
      IMPLICIT NONE
      CHARACTER*12 TARGET
      CHARACTER*80 SHIELDFILE
      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 INPUT_FLUX(NELM,MARR),OUTPUT_FLUX(NELM,MARR)
      REAL*4 TEMP_INPUT(NELM,MARR),TEMP_FLUX(NELM,MARR)
      REAL*4 ELOWER,EUPPER,UPATH0,PATH,PSTEP,PSTEPMIN,PSTEPMAX
      REAL*4 PATHOLD,DELTA_PATH,TEMP_PATH
      INTEGER*4 M,N,NSP,IZLO,IZUP,IPATH,IULABEL
      REAL*4 UPATH,UUPATH,FRACSHLD
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE
      INTEGER*4 MAXSHIELD,NSHIELD,K,IELM,IARR
      PARAMETER (MAXSHIELD=500)
      DIMENSION UPATH(MAXSHIELD),FRACSHLD(MAXSHIELD)
      CHARACTER*5 UNITS_LABEL
      DIMENSION UNITS_LABEL(4)
      DATA UNITS_LABEL/'g/cm2','mils ','cm   ','!!!!!'/
      INTEGER*4 IENT
      DATA IENT/0/
C
C
      WRITE(6,9998)
 9998 FORMAT(1x,' TRANSPORT_DRIVER calculation started.',
     &          '  Please stand by.')

      CALL GET_CREME96_VERSION(VERSION_NUMBER)
      PROGRAM_CODE=4
```

```
C
C      Now set parametes for transport calculation.
C      Use recommended default:

C      Special version: turns off all nuclear fragmention:
       N=0
       IF (IENT.EQ.0) THEN
           IENT=1
           WRITE(6,9995)
 9995      FORMAT(1x,'!!!! SPECIAL VERSION OF CREME96 TRANSPORT !!!!',
      &             /,' All nuclear fragmentation turned off!',
      &             /,' NOTE: The user is responsible for tracking',
      &               ' to which output files this pertains!')
       ENDIF

C      Use straight-ahead approximation; ignore energy spread of target fragments
C      (This takes a lot of time and generally has only very small effect.)
       NSP=0
C
C      Set maximum & minimum PSTEP sizes allowed in transport
       PSTEPMIN=0.20
       PSTEPMAX=0.20
       IULABEL=IPATH+1
       IF (IULABEL.GT.4) IULABEL=4


       IF (UPATH0.GT.0.0) THEN
           NSHIELD=1
           UPATH(1)=UPATH0
           FRACSHLD(1)=1.00
       ELSE
           CALL UNLOAD_SHIELDFILE(SHIELDFILE,NSHIELD,UPATH,FRACSHLD)
       ENDIF

       PATHOLD=0.0


C
       DO 1000 K=1,NSHIELD
       WRITE(6,999) K,UPATH(K),UNITS_LABEL(IULABEL),FRACSHLD(K)
 999   FORMAT(1x,' SHIELDING BIN ',I4,' THICKNESS = ',F10.4,1x,A5,
      &          '    FRACTION = ',F8.4)

       UUPATH=UPATH(K)

C      Get shielding thickness (PATH) in g/cm2 and transport step size:
       CALL UNLOAD_PATH(IPATH,UUPATH,TARGET,PATH,PSTEPMIN,PSTEPMAX,PSTEP)
C
C      Now perform transport:
       IF (NSHIELD.EQ.1) THEN
       CALL UPROP96(INPUT_FLUX,
      &                ELOWER,EUPPER,M,IZLO,IZUP,
      &                N,NSP,PATH,PSTEP,TARGET,
      &                OUTPUT_FLUX)
C
C
       ELSE
C
C      Modification 8-16-96 by AJT:
C      To speed up calculations through thick shielding distributions,
C      allow output of one step to be input to the next step.
```

```
      DELTA_PATH=PATH-PATHOLD
      IF (DELTA_PATH.LT.0.) DELTA_PATH=0.0
      DO 300 IELM=1,NELM
         DO 200 IARR=1,MARR
               IF (K.EQ.1) THEN
               TEMP_INPUT(IELM,IARR)=INPUT_FLUX(IELM,IARR)
               TEMP_PATH=PATH
               ELSEIF (K.GT.1 .and. DELTA_PATH.LT.PSTEP) THEN
               TEMP_INPUT(IELM,IARR)=INPUT_FLUX(IELM,IARR)
               TEMP_PATH=PATH
               ELSEIF (K.GT.1 .and. DELTA_PATH.GE.PSTEP) THEN
               TEMP_INPUT(IELM,IARR)=TEMP_FLUX(IELM,IARR)
               TEMP_PATH=DELTA_PATH
               ENDIF
  200    CONTINUE
  300 CONTINUE


      CALL UPROP96(TEMP_INPUT,
     &             ELOWER,EUPPER,M,IZLO,IZUP,
     &             N,NSP,TEMP_PATH,PSTEP,TARGET,
     &             TEMP_FLUX)
      PATHOLD=PATH

C     Now add to weighted sum:
      DO 500 IELM=1,NELM
         DO 400 IARR=1,MARR
            OUTPUT_FLUX(IELM,IARR)=OUTPUT_FLUX(IELM,IARR)+
     &                             TEMP_FLUX(IELM,IARR)*FRACSHLD(K)
  400    CONTINUE
  500 CONTINUE

      ENDIF

 1000 CONTINUE
C
      WRITE(6,9999)
 9999 FORMAT(1x,' TRANSPORT_DRIVER calculation completed.  Thank you.',
     & /,1x,' All fluxes are in units of particles/m2-s-sr-MeV/nuc',
     &      ' vs. energy in MeV/nuc.',
     & /,1x,' Recommended next step: ',
     & /,5x,' LETSPEC',
     &      ' (RUN CREME96:LETSPEC_DRIVER)'
     &      ' for heavy-ion induced SEUs;',
     & /,2x,' or PUP     (RUN CREME96:PROTON_UPSET_DRIVER)',
     &      ' for proton-induced SEUs.')


      RETURN
      END
```

```
      SUBROUTINE OUTPUT_CREME96_DIFLET(LETMIN,LETMAX,L,
     *                              IZLO,IZUP,EMINCUT,EMAXCUT,TARGET,
     *                              VERSION_NUMBER,PROGRAM_CODE,
     *                              INFILE,
     *                              DIFSPEC,
     *                              OUTFILE)
C
C     Routine for writing out the CREME96 differential LET Spectrum.
C
      IMPLICIT NONE
      REAL*4 LETMIN,LETMAX,EMINCUT,EMAXCUT,DIFSPEC
      DIMENSION DIFSPEC(1)
      INTEGER*4 L,IZLO,IZUP,LARR,K,NHEADER,NHEADER0,OUTUNIT
      DATA OUTUNIT/2/
      CHARACTER*12 TARGET
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME
      CHARACTER*80 INFILE,OUTFILE,DLTFILE
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE,STAT,CREME96_OPEN

C     FORMAT statements

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,16x,I4,1x,I1)
150   FORMAT(1x,A79)
200   FORMAT((1X,6(1PE10.4,2X)))
210   FORMAT((1X,6(E10.4,2X)))

C     Open output file and write header
C
C     First, modify name for DIFLET spectrum:
      DLTFILE='NULLFILE'
      DO K=2,LEN(OUTFILE)
            IF (OUTFILE(K:K) .EQ. '.') THEN
                DLTFILE=OUTFILE(1:K-1)//'.DLT'
            ENDIF
      ENDDO
      IF (DLTFILE.EQ.'NULLFILE') DLTFILE=OUTFILE//'.DLT'

c      OPEN(UNIT=OUTUNIT,STATUS='NEW',FILE='USER:'//DLTFILE)
      stat = creme96_open(dltfile,'user',outunit,'new')
      CALL DATE(CREATION_DATE)
      CALL TIME(CREATION_TIME)

      CALL CHECK_HEADER_LENGTH(INFILE,NHEADER0)
      NHEADER=NHEADER0+5
      WRITE(OUTUNIT,990) NHEADER,DLTFILE(1:70),
     &                   VERSION_NUMBER,PROGRAM_CODE+1
 990  FORMAT(I3,1x,A70,I4,1x,I1)
      WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME
 992  FORMAT(1x,'%Created by CREME96:LETSPEC_DRIVER Version ',I4,
     &          ' on ',A9,' at ',A8)
      WRITE(OUTUNIT,993) IZLO,IZUP,LETMIN,LETMAX,L
 993  FORMAT(1x,'%ZMIN =',I3,' ZMAX =',I3,
     &     ' LETMIN = ',1PE8.2,' LETMAX = ',1PE8.2,
     &     ' MeV-cm2/g LBINS =',I5)
      WRITE(OUTUNIT,994) EMINCUT
 994  FORMAT(1x,'%EMINCUT = ',1PE8.2,' MeV/nuc')
      WRITE(OUTUNIT,995) TARGET
 995  FORMAT(1x,'%TARGET MATERIAL = ',A12)
```

```
C     Now copy header information from input file:
      WRITE(OUTUNIT,998) INFILE(1:45)
 998  FORMAT(1x,'%Input File to LETSPEC_DRIVER: ',A45)
      CALL COPY_HEADERS(INFILE,NHEADER0,OUTUNIT)

C     Finally, output differential LET spectrum:

      WRITE(OUTUNIT,100) LETMIN,LETMAX,L,IZLO,IZUP,TARGET,
     &                VERSION_NUMBER,PROGRAM_CODE+1
      WRITE(OUTUNIT,100)

C     Write flux to file.

      WRITE(OUTUNIT,200) (DIFSPEC(K),K=1,L)
      CLOSE(OUTUNIT)

      RETURN
      END
```

```
      SUBROUTINE OUTPUT_CREME96_DOSE(INFILE,IZMIN,IZMAX,LETMIN,LETMAX,
     *                          EMINCUT,EMAXCUT,TARGET,MODEL_TYPE,
     *                          VERSION_NUMBER,PROGRAM_CODE,
     *                          DOSE_PER_SECOND,acc_dose,
     *                          OUTFILE)
C
C     Routine for writing out the CREME96 dose calculation.
C
      IMPLICIT NONE
      CHARACTER*80 INFILE,OUTFILE
      INTEGER*4 IZMIN,IZMAX,MODEL_TYPE
      REAL*4 LETMIN,LETMAX,EMINCUT,EMAXCUT
      REAL*4 DOSE_PER_SECOND,acc_dose
      INTEGER*4 NHEADER,NHEADER0,OUTUNIT
      DATA OUTUNIT/2/
      CHARACTER*12 TARGET
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE,STAT,CREME96_OPEN

C     Open output file and write header

c      OPEN(UNIT=OUTUNIT,STATUS='NEW',FILE='USER:'//OUTFILE)
      stat = creme96_open(outfile,'user',outunit,'new')
      CALL DATE(CREATION_DATE)
      CALL TIME(CREATION_TIME)

      CALL CHECK_HEADER_LENGTH(INFILE,NHEADER0)
      NHEADER=NHEADER0+6
      IF (MODEL_TYPE.EQ.1 .or. MODEL_TYPE.EQ.2) NHEADER=NHEADER+1
      WRITE(OUTUNIT,990) NHEADER,OUTFILE(1:70),
     &                   VERSION_NUMBER,PROGRAM_CODE
 990  FORMAT(I3,1x,A70,I4,I2)
      WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME
 992  FORMAT(1x,'%Created by CREME96:DOSE_DRIVER Version ',I4,
     &         ' on ',A9,' at ',A8)
      IF(MODEL_TYPE.EQ.0) WRITE(OUTUNIT,900) DOSE_PER_SECOND,acc_dose
      IF(MODEL_TYPE.EQ.1) WRITE(OUTUNIT,901) DOSE_PER_SECOND,acc_dose
      IF(MODEL_TYPE.EQ.2) WRITE(OUTUNIT,902) DOSE_PER_SECOND,acc_dose
      IF(MODEL_TYPE.EQ.3) WRITE(OUTUNIT,903) DOSE_PER_SECOND,acc_dose

 900  FORMAT(' %Average Dose = ',1PE13.6,' rad/sec = ',1PE13.6,
     &       ' krad/year')
 901  FORMAT(' %Worst-day average dose rate = ',1PE13.6,' rad/sec',
     & /,' %Event-Accumulated Dose = ',1PE13.6,' krad in 18.0 hours.')
 902  FORMAT(' %Worst-week average dose rate = ',E13.6,' rad/sec',
     & /,' %Event-Accumulated Dose = ',1PE13.6,' krad in 180.0 hours.')
 903  FORMAT(' %Peak SEP dose rate = '1P,E13.6,' rad/sec = ',
     &         1PE13.6,' krad/sec')

      WRITE(OUTUNIT,993) IZMIN,IZMAX,LETMIN,LETMAX
 993  FORMAT(1x,'%ZMIN =',I3,' ZMAX =',I3,
     &     ' LETMIN = ',1PE8.2,' LETMAX = ',1PE8.2,' MeV-cm2/g')
      WRITE(OUTUNIT,994) EMINCUT,EMAXCUT
 994  FORMAT(1x,'%EMINCUT = ',1PE8.2,' EMAXCUT = ',1PE8.2,' MeV/nuc')
      WRITE(OUTUNIT,995) TARGET
 995  FORMAT(1x,'%TARGET MATERIAL = ',A12)

C     Now copy header information from input file:
      WRITE(OUTUNIT,998) INFILE(1:45)
```

```
998   FORMAT(1x,'%Input File to DOSE_DRIVER: ',A45)
      CALL COPY_HEADERS(INFILE,NHEADER0,OUTUNIT)

      CLOSE(OUTUNIT)

      RETURN
      END
```

```
      SUBROUTINE OUTPUT_CREME96_FLUX(IZLO,IZHI,ELOWER,EUPPER,
     *                               YEAR,IMODE,ITRANS,
     *                               GTRANSFILE,TRAPDFILE,
     *                               VERSION_NUMBER,PROGRAM_CODE,
     *                               M,FLX,OUTFILE)
C
C     Routine for writing the CREME96 particle environment file.
C     Modified 7/29/96 to add tracking information to header.
C     Modified 8/19/96 to add more detailed header information,
C     per recommendation from Ed Petersen.
C
      IMPLICIT NONE
      INTEGER*4 IZLO,IZHI,J,K,M,OUTUNIT,STAT,CREME96_OPEN
      DATA OUTUNIT/2/
      INTEGER*4 NHEADER,NGTFLINES,NTRPLINES
      REAL*4 ELOWER,EUPPER
      CHARACTER*80 OUTFILE,ILINE
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME

      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 FLX
      DIMENSION FLX(NELM,MARR)
      REAL*4 YEAR
      INTEGER*4 IMODE,ITRANS,VERSION_NUMBER,PROGRAM_CODE
      CHARACTER*80 GTRANSFILE,TRAPDFILE
      CHARACTER*12 TARGET
      DATA TARGET/'UNSHIELDED  '/

C     FORMAT statements

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,
     &       2X,0PF8.3,1x,I2,1x,I1,1x,I4,1x,I1)
150   FORMAT(1x,A39,1x,A39)
200   FORMAT((1X,6(1PE10.4,2X)))

C     Open output file and write header

c      OPEN(UNIT=OUTUNIT,STATUS='NEW',FILE='USER:'//OUTFILE)
      stat = creme96_open(outfile,'user',outunit,'new')
      CALL DATE(CREATION_DATE)
      CALL TIME(CREATION_TIME)
C
C     Now prepare header for output file:
      NHEADER=3
      IF (ITRANS.EQ.0) NHEADER=NHEADER+1
      IF (ITRANS.GE.1) THEN
                 CALL CHECK_HEADER_LENGTH(GTRANSFILE,NGTFLINES)
                 NHEADER=NHEADER+2+NGTFLINES
      ENDIF
      IF (ITRANS.EQ.2) THEN
                 CALL CHECK_HEADER_LENGTH(TRAPDFILE,NTRPLINES)
                 NHEADER=NHEADER+1+NTRPLINES
      ENDIF

      WRITE(OUTUNIT,990) NHEADER,OUTFILE(1:70),
     &     VERSION_NUMBER,PROGRAM_CODE
990   FORMAT(I3,1x,A70,I4,1x,I1)
      WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME
```

```
 992  FORMAT(1x,'%Created by CREME96:FLUX_DRIVER Version ',I4,
     &           ' on ',A9,' at ',A8)

C       Revised 9/14/96: Energy limits hardwired
C       WRITE(OUTUNIT,993) IZLO,IZHI,ELOWER,EUPPER,M
C 993   FORMAT(1x,'%ZMIN =',I3,' ZMAX =',I3,
C      &      '  EMIN = ',1PE10.4,' EMAX = ',1PE10.4,
C      &      ' MeV/nuc  MBINS =',I5)

      WRITE(OUTUNIT,993) IZLO,IZHI
 993  FORMAT(1x,'%ZMIN =',I3,' ZMAX =',I3)

      IF (IMODE.EQ.0) THEN
         WRITE(OUTUNIT,995) IMODE,YEAR
 995     FORMAT(1x,'%IMODE  = ',I3,' SOLAR-QUIET MODE: YEAR = ',F10.4)
      ELSEIF (IMODE.EQ.1) THEN
         WRITE(OUTUNIT,996) IMODE
 996     FORMAT(1x,'%IMODE  = ',I3,
     &            ' WORST-DAY SOLAR ENERGETIC PARTICLE MODEL')
      ELSEIF (IMODE.EQ.2) THEN
         WRITE(OUTUNIT,997) IMODE
 997     FORMAT(1x,'%IMODE  = ',I3,
     &            ' WORST-WEEK SOLAR ENERGETIC PARTICLE MODEL')
      ELSE
         WRITE(OUTUNIT,998) IMODE
 998     FORMAT(1x,'%IMODE =',I3,' PEAK 5-MINUTE-AVERAGED FLUX')
      ENDIF

      IF (ITRANS.EQ.0) THEN
         WRITE(OUTUNIT,999) ITRANS
 999     FORMAT(1x,'%ITRANS = ',I3,
     &            ' GEOSYNCH/NEAR-EARTH INTERPLANETARY FLUXES')

      ELSEIF (ITRANS.EQ.1) THEN
         WRITE(OUTUNIT,899) ITRANS,GTRANSFILE(1:40)
 899     FORMAT(1x,'%ITRANS = ',I3,
     &            ' INSIDE MAGNETOSPHERE/NO TRAPPED FLUXES',
     &        /,1x,'%INPUT GEOMAGNETIC TRANSMISSION FILE:',1x,A40)
         CALL COPY_HEADERS(GTRANSFILE,NGTFLINES,OUTUNIT)

      ELSEIF (ITRANS.EQ.2) THEN
         WRITE(OUTUNIT,898) ITRANS,GTRANSFILE(1:40)
 898     FORMAT(1x,'%ITRANS = ',I3,
     &            ' INSIDE MAGNETOSPHERE/TRAPPED PROTONS INCLUDED',
     &       /,1x,'%INPUT GEOMAGNETIC TRANSMISSION FILE:',1x,A40)
         CALL COPY_HEADERS(GTRANSFILE,NGTFLINES,OUTUNIT)

         WRITE(OUTUNIT,897) TRAPDFILE(1:40)
 897     FORMAT(1x,'%INPUT TRAPPED PROTON FILE: ',11x,A40)
         CALL COPY_HEADERS(TRAPDFILE,NTRPLINES,OUTUNIT)
      ENDIF

      WRITE(OUTUNIT,100) ELOWER,EUPPER,M,IZLO,IZHI,
     &            TARGET,YEAR,IMODE,ITRANS,
     &            VERSION_NUMBER,PROGRAM_CODE
      WRITE(OUTUNIT,100)


C     Write fluxes to file.
```

```
      DO J=IZLO,IZHI
        WRITE(OUTUNIT,200) (FLX(J,K),K=1,M)
C       Skip line between elements.  AJT 5/6/96
        WRITE(OUTUNIT,100)
      END DO

      CLOSE(OUTUNIT)

      RETURN
      END
```

```
      SUBROUTINE OUTPUT_CREME96_LETSPEC(LETMIN,LETMAX,L,
     *                           IZLO,IZUP,EMINCUT,EMAXCUT,TARGET,
     *                           VERSION_NUMBER,PROGRAM_CODE,
     *                           INFILE,
     *                           SPECT,
     *                           OUTFILE)
C
C     Routine for writing out the CREME96 integral LET Spectrum.
C     Modified 7/29/96 to add header information.
C     Modified 8/19/96 to add more detailed header information, per
C     recommendation by Ed Petersen.
C
      IMPLICIT NONE
      REAL*4 LETMIN,LETMAX,EMINCUT,EMAXCUT
      INTEGER*4 L,IZLO,IZUP,LARR,K,NHEADER,NHEADER0,OUTUNIT
      DATA OUTUNIT/2/
      CHARACTER*12 TARGET
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME
      PARAMETER (LARR=1002)
      REAL*4 SPECT(LARR)
      CHARACTER*80 INFILE,OUTFILE
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE,STAT,CREME96_OPEN

C     FORMAT statements

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,16x,I4,1x,I1)
150   FORMAT(1x,A79)
200   FORMAT((1X,6(1PE10.4,2X)))

C     Open output file and write header

c      OPEN(UNIT=OUTUNIT,STATUS='NEW',FILE='USER:'//OUTFILE)
      stat = creme96_open(outfile,'user',outunit,'new')
      CALL DATE(CREATION_DATE)
      CALL TIME(CREATION_TIME)

      CALL CHECK_HEADER_LENGTH(INFILE,NHEADER0)
      NHEADER=NHEADER0+5
      WRITE(OUTUNIT,990) NHEADER,OUTFILE(1:70),
     &                   VERSION_NUMBER,PROGRAM_CODE
990   FORMAT(I3,1x,A70,I4,1x,I1)
      WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME
992   FORMAT(1x,'%Created by CREME96:LETSPEC_DRIVER Version ',I4,
     &          ' on ',A9,' at ',A8)
      WRITE(OUTUNIT,993) IZLO,IZUP,LETMIN,LETMAX,L
993   FORMAT(1x,'%ZMIN =',I3,' ZMAX =',I3,
     &      ' LETMIN = ',1PE8.2,' LETMAX = ',1PE8.2,
     &      ' MeV-cm2/g LBINS =',I5)
      WRITE(OUTUNIT,994) EMINCUT
994   FORMAT(1x,'%EMINCUT = ',1PE8.2,' MeV/nuc')
      WRITE(OUTUNIT,995) TARGET
995   FORMAT(1x,'%TARGET MATERIAL = ',A12)

C     Now copy header information from input file:
      WRITE(OUTUNIT,998) INFILE(1:45)
998   FORMAT(1x,'%Input File to LETSPEC_DRIVER: ',A45)
      CALL COPY_HEADERS(INFILE,NHEADER0,OUTUNIT)

C     Finally, output integral LET spectrum:
```

```
      WRITE(OUTUNIT,100) LETMIN,LETMAX,L,IZLO,IZUP,TARGET,
     &               VERSION_NUMBER,PROGRAM_CODE
      WRITE(OUTUNIT,100)

C     Write flux to file.

      WRITE(OUTUNIT,200) (SPECT(K),K=1,L)
      CLOSE(OUTUNIT)

      RETURN
      END
```

```fortran
      SUBROUTINE OUTPUT_SHIELDFILE(SHIELDFILE,
     *                             COMMENT,IUNITS,MATERIAL,
     *                             NBINS,XTHICK,XPROB,
     *                             XMEAN,XRMS,TOTAL,ERRFLAG,
     *                             VERSION_NUMBER,PROGRAM_CODE)
C
      IMPLICIT NONE
      CHARACTER*80 SHIELDFILE,COMMENT
      INTEGER*4 IUNITS,STAT,CREME96_OPEN
      CHARACTER*12 MATERIAL
      INTEGER*4 NBINS,VERSION_NUMBER,PROGRAM_CODE
      REAL*4 XTHICK,XPROB,XMEAN,XRMS,TOTAL
      INTEGER*4 ERRFLAG
      DIMENSION XTHICK(1),XPROB(1)
      INTEGER*4 NHEADER
      INTEGER*4 OUTUNIT,IULABEL,K
      DATA OUTUNIT/2/
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME
      CHARACTER*5 UNITS_LABEL
      DIMENSION UNITS_LABEL(4)
      DATA UNITS_LABEL/'g/cm2','mils ','cm   ','!!!!!'/

C     Open output file and write header

c      OPEN(UNIT=OUTUNIT,STATUS='NEW',FILE='USER:'//SHIELDFILE)
      stat = creme96_open(shieldfile,'user',outunit,'new')
      CALL DATE(CREATION_DATE)
      CALL TIME(CREATION_TIME)
      NHEADER=4
C
      WRITE(OUTUNIT,990) NHEADER,SHIELDFILE(1:70),
     &                   VERSION_NUMBER,PROGRAM_CODE
 990  FORMAT(I3,1x,A70,I4,1x,I1)
      WRITE(OUTUNIT,991) VERSION_NUMBER,CREATION_DATE,CREATION_TIME
 991  FORMAT(1x,'%Created by CREME96:SHIELDFILE_DRIVER Version ',I4,
     &          ' on ',A9,' at ',A8)
      WRITE(OUTUNIT,992) COMMENT(1:78)
 992  FORMAT(1X,'%',A78)
      IULABEL=IUNITS+1
      IF (IULABEL.GT.4) IULABEL=4
      WRITE(OUTUNIT,993) UNITS_LABEL(IULABEL),MATERIAL,NBINS
 993  FORMAT(1x,'%Shielding thicknesses: in ',A5,4x,A12,
     &                                 3x,' NBINS = ',I5)
      IF (ERRFLAG.EQ.0) WRITE(OUTUNIT,994) XMEAN,XRMS
 994  FORMAT(1x'%MEAN = ',E13.6,' RMS = ',E13.6)
      IF (ERRFLAG.EQ.1) WRITE(OUTUNIT,995) XMEAN,XRMS,TOTAL
 995  FORMAT(1x'%MEAN = ',E13.6,' RMS = ',E13.6,
     &          ' Sum Before Renorm = ',E13.6)
      WRITE(OUTUNIT,996) IUNITS
 996  FORMAT(I2)

      DO 1000 K=1,NBINS
         WRITE(OUTUNIT,999) XTHICK(K),XPROB(K)
 999     FORMAT(1x,E13.6,5x,E13.6)
1000  CONTINUE

      CLOSE(OUTUNIT)

      RETURN
```

END

```
      SUBROUTINE OUTPUT_TRANSPORTED_FLUX(IZLO,IZHI,ELOWER,EUPPER,
     *                                   IPATH,UPATH,TARGET,
     *                                   SHIELDFILE,INFILE,
     *                                   VERSION_NUMBER,PROGRAM_CODE,
     *                                   M,FLX,OUTFILE)
C
C     Routine for writing the CREME96 transported particle environment file.
C     Modified 7/29/96 to add tracking information to header.
C     Modified 8/19/96 to add more detailed header information, per
C     recommendation by Ed Petersen.
C
C
      IMPLICIT NONE
      INTEGER*4 IZLO,IZHI,J,K,M,NHEADER,NHEADER0,IULABEL
      INTEGER*4 OUTUNIT,STAT,CREME96_OPEN
      DATA OUTUNIT/2/
      REAL*4 ELOWER,EUPPER
      CHARACTER*80 OUTFILE,ILINE
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME
      CHARACTER*5 UNITS_LABEL
      DIMENSION UNITS_LABEL(4)
      DATA UNITS_LABEL/'g/cm2','mils ','cm   ','!!!!!'/

      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 FLX
      DIMENSION FLX(NELM,MARR)
      REAL*4 UPATH
      INTEGER*4 IPATH,IDUM,VERSION_NUMBER,PROGRAM_CODE,NSHDLINES
      DATA NSHDLINES/0/
      CHARACTER*80 INFILE,SHIELDFILE
      CHARACTER*12 TARGET
      DATA IDUM/0/

C     FORMAT statements

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,
     &       2X,0PF8.3,1x,I2,1x,I1,1x,I4,1x,I1)
150   FORMAT(1x,A39,1x,A39)
200   FORMAT((1X,6(1PE10.4,2X)))

C     Open output file and write header

c      OPEN(UNIT=OUTUNIT,STATUS='NEW',FILE='USER:'//OUTFILE)
      stat = creme96_open(outfile,'user',outunit,'new')
      CALL DATE(CREATION_DATE)
      CALL TIME(CREATION_TIME)
C
      CALL CHECK_HEADER_LENGTH(INFILE,NHEADER0)
      NHEADER=NHEADER0+4

      IF (SHIELDFILE(1:12).NE.'            ') THEN
          CALL CHECK_HEADER_LENGTH(SHIELDFILE,NSHDLINES)
          NHEADER=NHEADER+NSHDLINES
      ENDIF

      WRITE(OUTUNIT,990) NHEADER,OUTFILE(1:70),
     &                   VERSION_NUMBER,PROGRAM_CODE
990   FORMAT(I3,1x,A70,I4,1x,I1)
```

```
      WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME
 992  FORMAT(1x,'%Created by CREME96:TRANSPORT_DRIVER Version ',I4,
     &           ' on ',A9,' at ',A8)
      WRITE(OUTUNIT,993) IZLO,IZHI,ELOWER,EUPPER,M
 993  FORMAT(1x,'%ZMIN =',I3,' ZMAX =',I3,
     &      '   EMIN = ',1PE10.4,' EMAX = ',1PE10.4,
     &      ' MeV/nuc  MBINS =',I5)

      IULABEL=IPATH+1
      IF (IULABEL.GT.4) IULABEL=4

      IF (SHIELDFILE(1:12).NE.'            ') THEN
         WRITE(OUTUNIT,997) SHIELDFILE(1:50)
 997     FORMAT(1x,'%Shielding distribution: ',A50)
         CALL COPY_HEADERS(SHIELDFILE,NSHDLINES,OUTUNIT)
      ELSE
         WRITE(OUTUNIT,996) UPATH,UNITS_LABEL(IULABEL),TARGET
 996     FORMAT(1x,'%Thickness = ',F10.4,1x,A5,5x,A12)
      ENDIF

C     Now copy header information from input file:
      WRITE(OUTUNIT,998) INFILE(1:45)
 998  FORMAT(1x,'%Input File to TRANSPORT_DRIVER: ',A45)
      CALL COPY_HEADERS(INFILE,NHEADER0,OUTUNIT)

C     Finally, output transported spectra:

      WRITE(OUTUNIT,100) ELOWER,EUPPER,M,IZLO,IZHI,
     &                   TARGET,UPATH,IPATH,IDUM,
     &                   VERSION_NUMBER,PROGRAM_CODE
      WRITE(OUTUNIT,100)


C     Write fluxes to file.

      DO J=IZLO,IZHI
        WRITE(OUTUNIT,200) (FLX(J,K),K=1,M)
C       Skip line between elements.  AJT 5/6/96
        WRITE(OUTUNIT,100)
      END DO

      CLOSE(OUTUNIT)

      RETURN
      END
```

```
      SUBROUTINE PARTIALS(ENERGY,IZ,IA,NAME,MEAN_FREE_PATH,
     &  NSP,ELOSS,NORMALIZATION)
!-----------------------------------------------------------------
!       This program computes the charge changing mfps
!       for the nuclide (IZ,IA) on material NAME at ENERGY MeV/N.
!       The target nuclides are (JZ,JA) and the products are (KZ,KA)
!-----------------------------------------------------------------
c       INCLUDE 'CREME96:ZCOMMON.CMN'
        CHARACTER*12 NAME
        REAL NA(28),IADJ(28),NASPM(28),DENS,ETAD
        INTEGER NZ(28),IGAS,NAS
        COMMON /TBLOCK/DENS,ETAD,IGAS,NAS,
     &                 NZ,NA,IADJ,NASPM,
     &                 NTOTAL,AVGZ,AVGZ2,AVGA,AVGI

        REAL MEAN_FREE_PATH(100),CR(100),ELOSS(100),NORMALIZATION
        REAL IA,JA,KA
        DATA AVOGADRO/6.022045E23/    !  particles/mole
        CALL ZTARGET(NAME)
        FACT=1.E27*AVGA/AVOGADRO
        NORMALIZATION=0.
        DO KZ=3,IZ+1
          MEAN_FREE_PATH(KZ)=0.
          DO K=KZ,3*KZ
            KA=FLOAT(K)
            DO L=1,NAS
              JZ=NZ(L)
              JA=NA(L)
C
C.......The following modulates the projectile energy by the number of
C.......TARGET participants, July 1992:
              IF(NINT(JA).EQ.1) THEN
                E=AMIN1(ENERGY*JA,10000.)
              ELSE
                 AP=IA
                 AT=JA
                 CALL GLAUBER(AP,AT,AP_P,AT_P)
                 PART=AT_P+AP_P
                 DELTA=IA-KA
                 A_EFF=AT_P*(1.+TANH((DELTA-PART)/PART))
                 E=AMIN1(ENERGY*A_EFF,10000.)
              END IF
C
              CALL YIELDX(IZ,NINT(IA),KZ,NINT(KA),AMAX1(E,100.),S)
              IF (KZ.EQ.4.AND.K.EQ.8) S=0.
              IF (KZ.EQ.5.AND.K.EQ.9) S=0.
              IF (S.LT.1.E-4) S=0.
              CALL SCALER(IZ,NINT(IA),JZ,NINT(JA),KZ,NINT(KA),
     &        AMAX1(ENERGY,100.),SC)
              MEAN_FREE_PATH(KZ)=MEAN_FREE_PATH(KZ)+S*SC*NASPM(L)
              NORMALIZATION=NORMALIZATION+S*SC*NASPM(L)*KZ
            END DO
          END DO
          MEAN_FREE_PATH(KZ)=MEAN_FREE_PATH(KZ)/FACT/NTOTAL
        END DO
        MEAN_FREE_PATH(2)=0.
        DO K2=1,3
          IF (K2.EQ.1) KZ=2
          IF (K2.EQ.2) KZ=4
          IF (K2.EQ.3) KZ=5
```

```
          IF (K2.EQ.1) K=4
          IF (K2.EQ.2) K=8
          IF (K2.EQ.3) K=9
          KA=FLOAT(K)
          DO L=1,NAS
            JZ=NZ(L)
            JA=NA(L)
C.......The following modulates the projectile energy by the number of
C.......TARGET participants, July 1992:
              IF(NINT(JA).EQ.1) THEN
                E=AMIN1(ENERGY*JA,10000.)
              ELSE
                AP=IA
                AT=JA
                CALL GLAUBER(AP,AT,AP_P,AT_P)
                PART=AT_P+AP_P
                DELTA=IA-KA
                A_EFF=AT_P*(1.+TANH((DELTA-PART)/PART))
                E=AMIN1(ENERGY*A_EFF,10000.)
              END IF
C
            CALL YIELDX(IZ,NINT(IA),KZ,NINT(KA),AMAX1(E,100.),S)
            IF (KZ.EQ.4.OR.KZ.EQ.5) S=S*2.
            IF (S.LT.1.E-4) S=0.
            CALL SCALER(IZ,NINT(IA),JZ,NINT(JA),KZ,NINT(KA),
     &      AMAX1(ENERGY,100.),SC)
            MEAN_FREE_PATH(2)=MEAN_FREE_PATH(2)+S*SC*NASPM(L)
            NORMALIZATION=NORMALIZATION+S*SC*NASPM(L)*2.
          END DO
        END DO
        MEAN_FREE_PATH(2)=MEAN_FREE_PATH(2)/FACT/NTOTAL
        NORMALIZATION=NORMALIZATION/FACT/NTOTAL
C
C.......Energy-loss calculations: Sept. 1993
C
        IF(NSP.EQ. 0) RETURN
        DO KZ=1,IZ+1
          ELOSS(KZ)=0.
          CR(KZ)=0.
          DO K=KZ,3*KZ
            KA=FLOAT(K)
            DO L=1,NAS
              JZ=NZ(L)
              JA=NA(L)
C
C.......The following modulates the projectile energy by the number of
C.......TARGET participants, July 1992:
              IF(NINT(JA).EQ.1) THEN
                 E=AMIN1(ENERGY*JA,10000.)
              ELSE
                 AP=IA
                 AT=JA
                 CALL GLAUBER(AP,AT,AP_P,AT_P)
                 PART=AT_P+AP_P
                 DELTA=IA-KA
                 A_EFF=AT_P*(1.+TANH((DELTA-PART)/PART))
                 E=AMIN1(ENERGY*A_EFF,10000.)
              END IF
C
              CALL YIELDX(IZ,NINT(IA),KZ,NINT(KA),AMAX1(E,100.),S)
```

```
            IF (KZ.EQ.4.AND.K.EQ.8) S=0.
            IF (KZ.EQ.5.AND.K.EQ.9) S=0.
            IF (S.LT.1.E-4) S=0.
            CALL SCALER(IZ,NINT(IA),JZ,NINT(JA),KZ,NINT(KA),
     &      AMAX1(ENERGY,100.),SC)
            CALL E_LOSS(IZ,NINT(IA),JZ,NINT(JA),KZ,NINT(KA),
     &                  ENERGY,dKE,SigmaKE)
            dKE=dKE+0.*SigmaKE
            CR(KZ)=CR(KZ)+S*SC*NASPM(L)/NTOTAL
            IF(KZ.EQ.1) THEN
               ELOSS(KZ)=ELOSS(KZ)+dKE*(REAL(KA)/6.)*NASPM(L)
     &                   /NTOTAL
            ELSE
               ELOSS(KZ)=ELOSS(KZ)+dKE*S*SC*NASPM(L)/NTOTAL
            END IF
          END DO
        END DO
            IF(CR(KZ).NE.0.AND.KZ.GT.1) THEN
              ELOSS(KZ)=ELOSS(KZ)/CR(KZ)
            END IF
      END DO
C
      RETURN
      END
C
      SUBROUTINE SCALER(IZ1,IA1,IZ2,IA2,JZ,JA,E,SC)
      DATA IENT/0/
      IF (IENT.EQ.0) THEN
        IENT=1
C         TYPE *,' '
C         TYPE *,'    ^^^ Using STB scaling algorithm-1992 ^^^'
C         TYPE *,' '
      END IF
      SC = 1.
      IF (IA2.EQ.1)  RETURN
      SL = 1.
      S1 = 1.
      SD = 1.
      Z1 = IZ1
      A1 = IA1
      Z2 = IZ2
      A2 = IA2
      Z  = JZ
      A  = JA
      E1 = E/1000.
C     SC = 1.6 + 0.07*A2**(2./3.)
C
C.......New scaling algorithm:  July 1992
      SC=ASYMM(A1,A2)
C
      FA =(1.0 + (A1/120.)*AMIN1(E1,2.)/2.)/(1.+A1/120.)
      IF (JZ.LE.5)   SL = (1. + .4*(1.+.02*(Z1/Z)**2)*(1.-1.5*Z/Z1))*FA
      IF (A .LT.A1/2. .AND. JZ.GT.5)   SD = 3.*EXP(-(2.*A/A1))*FA
      IF (IA1-JA.EQ. 1)   S1 = (1. + .0014*Z1*Z2**(1.8-.005*Z2))/SC
      SC = SC*SL*S1*SD
      END
C
      SUBROUTINE GLAUBER(AP,AT,AP_P,AT_P)
C
C     calculates (average) number of proj. and target participants;
```

```
C       according to Glauber theory.
C
        DATA PI,R0/3.14159,1.36/
        DATA P13,P23/0.33333,0.66667/
C
        AP_P=AP * AT**P23 / (AP**P13+AT**P13)**2
        AT_P=AT * AP**P23 / (AP**P13+AT**P13)**2
        RETURN
        END
C
        FUNCTION ASYMM(AP,AT)
C       calculates asymmetry and participant contributions:
        DATA PI,R0/3.14159,1.36/
        DATA P13,P23/0.33333,0.66667/
C
        CALL GLAUBER(AP,AT,AP_P,AT_P)
C
        EXPO=EXP(-(AP-AT)/(AT+AP))
        RHO=(AP-1.)/(AP+1.)
        CONST=EXP(-RHO)                   !Normalization constant.
        ASYMM=CONST*(AP/AP_P)*EXPO
        RETURN
        END
```

```
c*********************************************************************
c
c       Module: PC_ROUTINES
c
c
c       Logical Names and Environment Variables serve the same purpose,
c       but are handled differently, on the two CREME96 platforms (VAX and
c       PC respectively).  There are also differences between the two file
c       OPEN statements.  To enable platform independance where fully
c       specified filenames and where file opens are used in the higher
c       level CREME96 code, two versions exist of the routines to perform
c       these tasks.  When an executable is being created, it is the
c       responsibility of the person performing the link to ensure that the
c       appropriate set of routines is used for the current build.
c
c
c       Two platform-DEPENDANT routines exist:
c
c               CREME96_FULL_FILENAME   creates fully-specified filename
c               CREME96_OPEN            performs a file OPEN on full filename
C               SHOW_DIRECTORY          gives advice when file not found.
c
c       These routines reside in the following 2 physical files:
c
c               VAX_ROUTINES.FOR        used for a VAX build
c               PC_ROUTINES.FOR         used for a PC build (this file)
c
c*********************************************************************


        integer function creme96_open(filename,path,unit,status)


c       FILENAME:       The non-fully specified name of the target file.
c
c       PATH:           Contains the DOS Environment Variable  pointing to
c                       the directory where file does, or will exist.
c
c       UNIT:           The logical unit to be associated with the file.
c                       Must be defined at the time of the function call
c                       (one will not be assinged by this routine).
c
c       STATUS:         Contains either OLD, for existing file, or
c                       NEW, to create a file.
c
c       Calling example:
c
c          STAT = creme96_open('input.dat','creme96',inunit,'old')
c
c       Success is indicated by a ZERO return value.  Otherwise, the
c       return value will contain the FORTRAN error code.
c
c
        IMPLICIT NONE
        character*80    filename,file,full_filename,line
        character*10    path
        integer         unit,ios
        character*7     status
```

```
      file = full_filename(filename,path)

      if (status(1:1) .eq. 'o' .or. status(1:1) .eq. 'O') then
c        Old files are only opened for READ (no APPEND in CREME).
c        Any file opened for READ will be opened SHARED.

         OPEN(UNIT=unit,file=file,status='old',
     &   mode='read',share='denywr',iostat=ios,err=199)

c DEBUG
c        read(unit,99)line
c99      format(a80)
c        write(*,*)'First line in file: ',line

      else
c        New file to be created.  WRITE and NOSHARE are default.  On the PC,
c        we must open with REPLACE instead of NEW, in case a file already
c        exists of this name (as it is our intention to write over it).  If
c        one doesn't exist, REPLACE acts the same as NEW.

         OPEN(UNIT=unit,file=file,status='replace',
     &   iostat=ios,err=199)

c DEBUG
c        write(*,*)'Writing test line to new file...'
c        write(unit,*)'Test line'

      endif

199   creme96_open_file = ios
      return
      end
c
c**********************************************************************
c
      character*80 function full_filename(filename,path)

      use msflib
      IMPLICIT NONE
      character*80    filename,dir
      character*10    path
      integer         lendir

c     The variable PATH contains the name of the environment variable
c     which in turn points to the directory path of the target file.
c     The function GETENVQQ will translate this environment variable
c     into the program variable DIR.

      lendir = getenvqq(path(1:len_trim(path)),dir)

      full_filename = dir(1:lendir)//'\'//filename

      return
      end
c
c**********************************************************************
c
      SUBROUTINE SHOW_DIRECTORY(JFILETYPE)
C
C     PC version.
```

```
C
C     The VAX version of this routine uses LIB$SPAWN to echo back a copy
C     of the user's directory when this routine becomes activated (because
C     the requested file was not found, etc. )  However, no comparable
C     capability exists on the PC.  We therefore just print out statements
C     which recommend that the user open another window and check his/her
C     directory.
C
      INTEGER*4 JFILETYPE

      IF (JFILETYPE.EQ.0) THEN
       WRITE(6,9010)
 9010  FORMAT(1x,' Please open another window and',
     & ' check the directory of your current USER area:')

      ELSEIF (JFILETYPE.EQ.1) THEN
       WRITE(6,9011)
 9011  FORMAT(1x,' Please open another window and',
     & ' check the directory of your *.tr* files.')

      ELSEIF (JFILETYPE.EQ.2) THEN
       WRITE(6,9012)
 9012  FORMAT(1x,' Please open another window and',
     & ' check the directory of your *.gt* files.')

      ELSEIF (JFILETYPE.EQ.3) THEN
       WRITE(6,9013)
 9013  FORMAT(1x,' Please open another window and',
     & ' check the directory of your particle flux files:',/,
     & ' *.flx, *.tfx, *.tr*')

      ELSEIF (JFILETYPE.EQ.4) THEN
       WRITE(6,9014)
 9014  FORMAT(1x,' Please open another windw and',
     & ' check the directory of your particle flux files:',/,
     & ' *.tfx, *.flx, *.tr*')

      ELSEIF (JFILETYPE.EQ.5) THEN
       WRITE(6,9015)
 9015  FORMAT(1x,' Please open another window and',
     & ' check the directory of your *.LET files.')

      ELSEIF (JFILETYPE.EQ.6) THEN
       WRITE(6,9016)
 9016  FORMAT(1x,' Please open another winow and',
     & ' check the directory of your *.DLT files.')

      ELSEIF (JFILETYPE.EQ.7) THEN
       WRITE(6,9017)
 9017  FORMAT(1x,' Please open another window and',
     & ' check the directory of your *.SHD files.')

      ELSEIF (JFILETYPE.EQ.8) THEN
       WRITE(6,9018)
 9018  FORMAT(1x,' Please open another window and',
     & ' check the directory of your *.XSD files.')

      ENDIF

      WRITE(6,9999)
```

```
9999  FORMAT(/)
      RETURN
      END
```

```
      SUBROUTINE PROTON_UPSETS(PROTON_FILE,IPARAM,PARAMS,
     &                         XSECT_FILE,NBITS,IENTER,
     &                         SEU_RATE,DAY_RATE,PERSECOND,PERDAY)
C
C     Subroutine for performing proton SEU evaluation:
C
C     Inputs:  PROTON_FILE = file containing proton differential flux
C                            (in protons/m2-s-sr-MeV) vs. energy (in MeV)
C              IPARAM      = 1,2,4, indicating cross-section model
C                            1 = Bendel 1-parameter
C                            2 = Bendel 2-parameter
C                            4 = Weibull
C                            0 = table
C              PARAMS(4)     = array containing cross-section parameters
C              XSECT_FILE = file containing cross-section table.
C              IENTER     = steering flag, for multiple calculations with
C                           the same proton spectrum.
C
C     Outputs:  SEU_RATE    in SEUs/s/bit
C               DAY_RATE    in SEUs/bit/day
C               PERSECOND   in SEUs/device/second
C               PERDAY      in SEUs/device/day
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C     Last update:   20 August 1996
C
C-----------------------------------------------------------------------
C
      IMPLICIT NONE
      INTEGER*4 NBINS,NPTS,IPARAM,IENTER
      REAL*4 EN,FLUX,PARAMS,XSECT,NBITS,
     &       SEU_RATE,DAY_RATE,PERSECOND,PERDAY
      CHARACTER*80 PROTON_FILE,XSECT_FILE

      PARAMETER (NBINS=5000)
      DIMENSION EN(NBINS),FLUX(NBINS),XSECT(NBINS)
      DIMENSION PARAMS(4)

      WRITE(6,9998)
 9998 FORMAT(1x,' PROTON_UPSET_DRIVER calculation started.',
     &          '  Please stand by.')


      SEU_RATE=0.0

C
C     On first entry, get proton spectrum:

      IF (IENTER.EQ.1) THEN

      CALL UNLOAD_PROTON_SPECTRUM(PROTON_FILE,EN,FLUX,NPTS)

      ENDIF
```

```
C       Evaluate proton SEU cross-section at these energy values:

        CALL EVALUATE_SEU_CROSS_SECTION(EN,NPTS,IPARAM,PARAMS,
     &                                  XSECT_FILE,XSECT)

C       Calculate SEU rate:

        CALL INTEGRATE_PROTON_UPSETS(NPTS,EN,FLUX,XSECT,SEU_RATE)


        IF (SEU_RATE.LT.0.) THEN
            WRITE(6,999) SEU_RATE
 999        FORMAT(1x,' ERROR in PROTON_UPSETS: SEU RATE = ',E13.5)
            SEU_RATE=0.0
        ENDIF

        CALL CALC_SEU_RATE(NBITS,SEU_RATE,DAY_RATE,PERSECOND,PERDAY)

        WRITE(6,9999)
 9999   FORMAT(1x,' PROTON_UPSET_DRIVER calculation completed. ')


        RETURN
        END
```

```
      PROGRAM PROTON_UPSET_DRIVER

      IMPLICIT NONE
      REAL*4 NBITS,PARAMS,SEU_RATE,DAY_RATE,PERSECOND,PERDAY
      REAL*4 XDUM
      INTEGER*4 IPARAM,IREPEAT,IENTER
      DIMENSION PARAMS(4)
      CHARACTER*80 PROTON_FILE,XSECT_FILE,REPORT_FILE
      CHARACTER*40 DEVICE_LABEL
      INTEGER*4 IERR
      DATA IERR/0/

      IENTER=1


10    CONTINUE
      CALL INITIALIZE_PROTON_UPSETS(PROTON_FILE,NBITS,
     &                              IPARAM,PARAMS,XSECT_FILE,IENTER,
     &                              DEVICE_LABEL,REPORT_FILE)


      CALL PROTON_UPSETS(PROTON_FILE,IPARAM,PARAMS,XSECT_FILE,
     &                   NBITS,IENTER,
     &                   SEU_RATE,DAY_RATE,PERSECOND,PERDAY)


      CALL PROTON_UPSET_REPORT(PROTON_FILE,NBITS,
     &                         IPARAM,PARAMS,XSECT_FILE,IENTER,
     &                         DEVICE_LABEL,REPORT_FILE,
     &                         SEU_RATE,DAY_RATE,PERSECOND,PERDAY)

 9100    CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(6,9200)
 9200    FORMAT(//,' Repeat SEU rate calculation with different',
     &             ' device characteristics? (1=yes,0=no)')
         READ(*,*,ERR=9100,IOSTAT=IERR) IREPEAT
         IF (IREPEAT.EQ.1) THEN
             IENTER=IENTER+1
             GOTO 10
         ENDIF

       WRITE(6,9600)
 9600  FORMAT(1x,' Proton Upset calculations finished.')

       STOP
       END
```

```fortran
      SUBROUTINE PROTON_UPSET_REPORT(PROTON_FILE,NBITS,
     &                          IPARAM,PARAMS,XSECT_FILE,IENTER,
     &                          DEVICE_LABEL,REPORT_FILE,
     &                          SEU_RATE,DAY_RATE,PERSECOND,PERDAY)


      IMPLICIT NONE
      REAL*4 NBITS,PARAMS,SEU_RATE,DAY_RATE,PERSECOND,PERDAY
      INTEGER*4 IPARAM,IENTER,OUTUNIT,VERSION_NUMBER,NHEADER0,K
      INTEGER*4 NHEADER,PROGRAM_CODE,STAT,CREME96_OPEN
      DATA OUTUNIT/2/
      DIMENSION PARAMS(4)
      CHARACTER*80 PROTON_FILE,XSECT_FILE,REPORT_FILE
      CHARACTER*40 DEVICE_LABEL
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME

      PROGRAM_CODE=9

      IF (IENTER.EQ.1.and.REPORT_FILE.NE.'NULLFILE') THEN
C        OPEN(UNIT=OUTUNIT,FILE='USER:'//REPORT_FILE,STATUS='NEW')
       stat = creme96_open(report_file,'user',outunit,'new')
       CALL DATE(CREATION_DATE)
       CALL TIME(CREATION_TIME)
       CALL GET_CREME96_VERSION(VERSION_NUMBER)
       CALL CHECK_HEADER_LENGTH(PROTON_FILE,NHEADER0)
       NHEADER=NHEADER0+2
       WRITE(OUTUNIT,991) NHEADER,REPORT_FILE(1:70),
     &                    VERSION_NUMBER,PROGRAM_CODE
 991   FORMAT(I3,1x,A70,I4,I2)
       WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME
 992   FORMAT(1x,'%Created by CREME96:PROTON_UPSET_DRIVER Version ',I4,
     &          ' on ',A9,' at ',A8)

C      Now copy header information from input file:
       WRITE(OUTUNIT,993) PROTON_FILE(1:40)
 993   FORMAT(1x,'%Input Proton Spectrum File: ',A40)
       CALL COPY_HEADERS(PROTON_FILE,NHEADER0,OUTUNIT)
      ENDIF

      IF (REPORT_FILE.NE.'NULLFILE') THEN
       WRITE(OUTUNIT,994) IENTER,DEVICE_LABEL
 994   FORMAT(/,1x,' REPORT NO. ',I4,': ',2x,A40)
       IF (IPARAM.EQ.0) WRITE(outunit,980) IPARAM,XSECT_FILE(1:75)
       IF (IPARAM.EQ.1) WRITE(outunit,981) IPARAM,PARAMS(1)
       IF (IPARAM.EQ.2) WRITE(outunit,982) IPARAM,PARAMS(1),PARAMS(2)
       IF (IPARAM.EQ.4) WRITE(outunit,984) IPARAM,(PARAMS(K),K=1,4)
       WRITE(outunit,996) NBITS
 996   FORMAT(1x,' Number of bits = ',E13.5)

 980   FORMAT(1x,' CROSS-SECTION INPUT ',I3,'  FROM FILE: ',
     &       /,5x,A75)
 981   FORMAT(1x,' CROSS-SECTION INPUT ',I3,
     &            '  BENDEL 1-PARAMETER = ',E13.5)
 982   FORMAT(1x,' CROSS-SECTION INPUT ',I3,
     &            '  BENDEL 2-PARAMETERS A,B = ',2E13.5)
 984   FORMAT(1x,' CROSS-SECTION INPUT ',I3,
     &            '  WEIBULL FIT: ',
     &       /,5x,' ONSET   = ',F9.3,' MeV',
     &       /,5x,' WIDTH   = ',F9.3,' MeV',
```

```
    &        /,5x,' POWER    = ',F9.3,' (dimensionless)',
    &        /,5x,' PLATEAU = ',F9.3,' x 10**-12 cm2/bit')


       WRITE(outunit,9200)
       WRITE(outunit,9201) IENTER,SEU_RATE,DAY_RATE,PERSECOND,PERDAY
9200   FORMAT(2x,'Rates:      SEUs/bit/second     /bit/day',
    &             '    /device/second    /device/day')
9201   FORMAT(2x,'*****',I4,2x,4(E14.5,2x))
       ENDIF

       WRITE(6,9200)
       WRITE(6,9201) IENTER,SEU_RATE,DAY_RATE,PERSECOND,PERDAY

      RETURN
      END
```

```
      SUBROUTINE RANGE(E,N,Z0,A1,NAME,R)
C     ************************************************************
C     *  THIS PROGRAM TABULATES THE RANGE OF NUCLIDE (Z0,A1) IN
C     *  A STOPPING MEDIUM 'NAME' AT ENERGIES GIVEN IN THE ARRAY
C     *  E IN MeV/nucleon.
C     ************************************************************
      CHARACTER*12 NAME
      DIMENSION E(N),R(N),GX(4),GA(4)
      COMMON AVGZ,AVGZ2,AVGA,AVGI
C     DATA NGAUSS/8/
C     DATA GX/1.96028986,1.79666648,1.52553241,1.18343464,
C    &        0.81656536,0.47446759,0.20333352,0.03971014/
C     DATA GA/0.10122854,0.22238103,0.31370665,0.36268378,
C    &        0.36268378,0.31370665,0.22238103,0.10122854/
      DATA NGAUSS/4/
      DATA GX/1.86113631,1.33998104,0.66001896,0.13886369/
      DATA GA/0.34785485,0.65214515,0.65214515,0.34785485/

      SS=STPOW(E(1),Z0,A1,NAME)
      JTEST=0
      DO J=1,N
        IF (E(J).LE.0.) THEN
          R(J)=0.
        ELSE
          IF (JTEST.EQ.0) THEN
            ELAST=0.
            RLAST=0.
            JTEST=1
          ELSE
            ELAST=E(J-1)
            RLAST=R(J-1)
          ENDIF
          DE=(E(J)-ELAST)/2.
          R(J)=0.
          DO K=1,NGAUSS
            STEMP=STPOW(ELAST+DE*GX(K),Z0,A1,NAME)
            IF (STEMP.GT.0.) R(J)=R(J)+GA(K)/STEMP
          END DO
          R(J)=DE*R(J)+RLAST
        ENDIF
      END DO
      RETURN
      END
```

```
      SUBROUTINE RETRY_INPUT(IERR)
C
C     NOTE:  A non-zero input value of IERR will be re-set to zero
C     by this routine.
C
      IMPLICIT NONE
      INTEGER*4 IERR
      LOGICAL RETRY
C     Logical flag RETRY may be set to .FALSE. to suppress repitition
C     of question after an incorrect response.  In this case, an
C     error message is printed and execution is terminated.  This feature
C     may be useful in the WWW version of the code, which is not truly
C     interactive.
      DATA RETRY/.TRUE./
C
      IF (IERR.NE.0) THEN
          IF (.NOT.RETRY) THEN
               WRITE(6,667)
 667           FORMAT('@ 00001 ABNORMAL TERMINATION: ',
     &         /,1x,' ERROR IN RETRY_INPUT: ',
     &         /,1x,' ERROR in user-supplied input.  STOP')
               STOP
          ELSE
               WRITE(6,665) IERR
 665           FORMAT(/,' ERROR ON INPUT: VAX ERROR CODE = ',I5,
     &              ' PLEASE TRY AGAIN.')
               IERR=0
          ENDIF
      ENDIF

      RETURN
      END
```

```
      REAL FUNCTION SEP_FLUX(IZ,EN,IMODE)
C
C     Returns the interplanetary Solar Energetic Particle differential flux
C     for element IZ, energy EN (MeV/nuc) in one of two modes:
C
C     IMODE=1: "Worst Day": particle fluxes based on observations from
C              GOES (protons) and IMP-8 (heavy ions) for the 18-hour
C              period beginning at 1300 UT on 20 OCT 1989.
C     IMODE=2: "Worst Week": particle fluxes based on observations from
C              GOES (protons) and IMP-8 (heavy ions) for the 180-hour
C              period beginning at 1300 UT on 19 OCT 1989.
C              NOTE:  The actual termination time here is somewhat arbitrary,
C                     since only a few percent of the flux was accumulated
C                     during the last day.
C     IMODE=3: Peak flux, based on peak 5-minute average flux observed on
C              GOES in October 1989
C
C
C      Average particle flux for this specified period is returned in
C      units of ions/m2-s-sr-MeV/nuc.
C
       IMPLICIT NONE
       INTEGER*4 IZ,IMODE,NSEP,K,IUSE
       REAL*4 EN,HOURS,SOLAR_PROTONS,ERRFLUX,SOLAR_HEAVY_IONS
       REAL*4 ETEMP
       CHARACTER*7 LSEP,LABEL
       DIMENSION HOURS(3),NSEP(2),LSEP(4)
       DATA HOURS/18.0,180.0,0.083333/
       DATA NSEP/1,4/
       DATA LSEP/'20OCT89','19OCT89','22OCT89','24OCT89'/

       SEP_FLUX=0.0
       IF (EN.LT.1.0 .or. EN.GT. 1.0E+5) RETURN
       IF (IMODE.LT.1 .or. IMODE.GT.3) RETURN

       IF (IZ.EQ.1) THEN
          ETEMP=EN
          IF (ETEMP.GT.600.) ETEMP=600.
          IF (IMODE.LT.3) THEN
          DO 1000 K=1,NSEP(IMODE)
            LABEL=LSEP(K)
            SEP_FLUX=SEP_FLUX+SOLAR_PROTONS(ETEMP,LABEL,ERRFLUX)
 1000     CONTINUE

          ELSEIF (IMODE.EQ.3) THEN
            LABEL='PEAKFLX'
            SEP_FLUX=SOLAR_PROTONS(ETEMP,LABEL,ERRFLUX)
          ENDIF

C         Match high-energy extrapolation to alpha spectrum; AJT 12/27/96

          IF (IMODE.NE.2) SEP_FLUX=SEP_FLUX*(ETEMP/EN)**4.14060
          IF (IMODE.EQ.2) SEP_FLUX=SEP_FLUX*(ETEMP/EN)**3.76100

        ELSEIF (IZ.EQ.2) THEN
            IUSE=6
C           Use D. Reames nominal He/C ratio:
            SEP_FLUX=122.5*SOLAR_HEAVY_IONS(IUSE,EN,IMODE,ERRFLUX)
C           Fine-tune using the GOES alpha flux at ~7.3 MeV/nuc
            IF (IMODE.NE.2) SEP_FLUX=0.85*SEP_FLUX
```

```
          IF (IMODE.EQ.2) SEP_FLUX=0.70*SEP_FLUX
      ELSE

          SEP_FLUX=SOLAR_HEAVY_IONS(IZ,EN,IMODE,ERRFLUX)
      ENDIF

C
C     Fluence evaluation completed.  Now normalize by elapsed time:
C
      SEP_FLUX=SEP_FLUX/(HOURS(IMODE)*3600.)
      IF (SEP_FLUX.LT.0.) SEP_FLUX=0.0
      RETURN
      END
```

```
C SHELLIG.FOR, Version 2.0, January 1992
C
C 11/1/91 SHELLG: lowest starting point for B0 search is 2
C 1/27/92 Adopted to IGRF-91 coeffcients model
C 2/5/92  Reduce variable-names: INTER(P)SHC,EXTRA(P)SHC,INITI(ALI)ZE
C 12/9/92 DGRF/IGRF file names changed by AJT
C         Changes in FELDCOF, for initialization purposes
C
C*********************************************************************
C  SUBROUTINES FINDB0, SHELLG, STOER, FELDG, FELDCOF, GETSHC,        *
C       INTERSHC, EXTRASHC, INITIZE                                  *
C*********************************************************************
C*********************************************************************
C
C
      SUBROUTINE FINDB0(STPS,BDEL,VALUE,BEQU,RR0)
C--------------------------------------------------------------------
C FINDS SMALLEST MAGNETIC FIELD STRENGTH ON FIELD LINE
C
C INPUT:   STPS   STEP SIZE FOR FIELD LINE TRACING
C       COMMON/FIDB0/
C          SP     DIPOLE ORIENTED COORDINATES FORM SHELLG; P(1,*),
C                 P(2,*), P(3,*) CLOSEST TO MAGNETIC EQUATOR
C          BDEL   REQUIRED ACCURACY  = [ B(LAST) - BEQU ] / BEQU
C                 B(LAST)  IS FIELD STRENGTH BEFORE BEQU
C
C OUTPUT:  VALUE  =.FALSE., IF BEQU IS NOT MINIMAL VALUE ON FIELD LINE
C          BEQU   MAGNETIC FIELD STRENGTH AT MAGNETIC EQUATOR
C          RR0    EQUATORIAL RADIUS NORMALIZED TO EARTH RADIUS
C          BDEL   FINAL ACHIEVED ACCURACY
C--------------------------------------------------------------------
      DIMENSION         P(8,4),SP(3)
      LOGICAL           VALUE
      COMMON/FIDB0/     SP
C
      STEP=STPS
      IRUN=0
7777  IRUN=IRUN+1
      IF(IRUN.GT.5) THEN
        VALUE=.FALSE.
        GOTO 8888
        ENDIF
C*******************FIRST THREE POINTS
      P(1,2)=SP(1)
      P(2,2)=SP(2)
      P(3,2)=SP(3)
      STEP=-SIGN(STEP,P(3,2))
      CALL STOER(P(1,2),BQ2,R2)
      P(1,3)=P(1,2)+0.5*STEP*P(4,2)
      P(2,3)=P(2,2)+0.5*STEP*P(5,2)
      P(3,3)=P(3,2)+0.5*STEP
      CALL STOER(P(1,3),BQ3,R3)
      P(1,1)=P(1,2)-STEP*(2.*P(4,2)-P(4,3))
      P(2,1)=P(2,2)-STEP*(2.*P(5,2)-P(5,3))
      P(3,1)=P(3,2)-STEP
      CALL STOER(P(1,1),BQ1,R1)
      P(1,3)=P(1,2)+STEP*(20.*P(4,3)-3.*P(4,2)+P(4,1))/18.
      P(2,3)=P(2,2)+STEP*(20.*P(5,3)-3.*P(5,2)+P(5,1))/18.
      P(3,3)=P(3,2)+STEP
      CALL STOER(P(1,3),BQ3,R3)
```

```
C******************INVERT SENSE IF REQUIRED
      IF(BQ3.LE.BQ1) GOTO 2
        STEP=-STEP
        R3=R1
        BQ3=BQ1
        DO 1 I=1,5
                ZZ=P(I,1)
                P(I,1)=P(I,3)
1               P(I,3)=ZZ
C******************INITIALIZATION
2       STEP12=STEP/12.
        VALUE=.TRUE.
        BMIN=1.E4
        BOLD=1.E4
C******************CORRECTOR (FIELD LINE TRACING)
        N=0
5555  P(1,3)=P(1,2)+STEP12*(5.*P(4,3)+8.*P(4,2)-P(4,1))
        N=N+1
      P(2,3)=P(2,2)+STEP12*(5.*P(5,3)+8.*P(5,2)-P(5,1))
C******************PREDICTOR (FIELD LINE TRACING)
      P(1,4)=P(1,3)+STEP12*(23.*P(4,3)-16.*P(4,2)+5.*P(4,1))
      P(2,4)=P(2,3)+STEP12*(23.*P(5,3)-16.*P(5,2)+5.*P(5,1))
      P(3,4)=P(3,3)+STEP
      CALL STOER(P(1,4),BQ3,R3)
        DO 1111 J=1,3
        DO 1111 I=1,8
1111    P(I,J)=P(I,J+1)
        B=SQRT(BQ3)
        IF(B.LT.BMIN) BMIN=B
        IF(B.LE.BOLD) THEN
                BOLD=B
                ROLD=1./R3
                SP(1)=P(1,4)
                SP(2)=P(2,4)
                SP(3)=P(3,4)
                GOTO 5555
                ENDIF
        IF(BOLD.NE.BMIN) THEN
                VALUE=.FALSE.
                ENDIF
        BDELTA=(B-BOLD)/BOLD
        IF(BDELTA.GT.BDEL) THEN
                STEP=STEP/10.
                GOTO 7777
                ENDIF
8888    RR0=ROLD
        BEQU=BOLD
        BDEL=BDELTA
        RETURN
        END
C
C
      SUBROUTINE SHELLG(GLAT,GLON,ALT,DIMO,FL,ICODE,B0)
C-----------------------------------------------------------------------
C CALCULATES L-VALUE FOR SPECIFIED GEODAETIC COORDINATES, ALTITUDE
C AND GEMAGNETIC FIELD MODEL.
C REF: G. KLUGE, EUROPEAN SPACE OPERATIONS CENTER, INTERNAL NOTE
C      NO. 67, 1970.
C      G. KLUGE, COMPUTER PHYSICS COMMUNICATIONS 3, 31-35, 1972
C-----------------------------------------------------------------------
```

```
C CHANGES (D. BILITZA, NOV 87):
C   - USING CORRECT DIPOL MOMENT I.E.,DIFFERENT COMMON/MODEL/
C   - USING IGRF EARTH MAGNETIC FIELD MODELS FROM 1945 TO 1990
C-----------------------------------------------------------------------
C  INPUT:  ENTRY POINT SHELLG
C               GLAT  GEODETIC LATITUDE IN DEGREES (NORTH)
C               GLON  GEODETIC LONGITUDE IN DEGREES (EAST)
C               ALT   ALTITUDE IN KM ABOVE SEA LEVEL
C
C          ENTRY POINT SHELLC
C               V(3)  CARTESIAN COORDINATES IN EARTH RADII (6371.2 KM)
C                       X-AXIS POINTING TO EQUATOR AT 0 LONGITUDE
C                       Y-AXIS POINTING TO EQUATOR AT 90 LONG.
C                       Z-AXIS POINTING TO NORTH POLE
C
C          DIMO       DIPOL MOMENT IN GAUSS (NORMALIZED TO EARTH RADIUS)
C
C          COMMON
C               X(3)    NOT USED
C               H(144)  FIELD MODEL COEFFICIENTS ADJUSTED FOR SHELLG
C-----------------------------------------------------------------------
C  OUTPUT: FL           L-VALUE
C          ICODE        =1 NORMAL COMPLETION
C                       =2 UNPHYSICAL CONJUGATE POINT (FL MEANINGLESS)
C                       =3 SHELL PARAMETER GREATER THAN LIMIT UP TO
C                          WHICH ACCURATE CALCULATION IS REQUIRED;
C                          APPROXIMATION IS USED.
C          B0           MAGNETIC FIELD STRENGTH IN GAUSS
C-----------------------------------------------------------------------
      DIMENSION         V(3),U(3,3),P(8,100),SP(3)
C     The following was an unlabeled common, which I have appropriately
C     named.  AJT 12-30-92.
      COMMON/BASTARDS/          X(3),H(144)
      COMMON/FIDB0/     SP
      COMMON/GENER/     UMR,ERA,AQUAD,BQUAD
C
C-- RMIN, RMAX ARE BOUNDARIES FOR IDENTIFICATION OF ICODE=2 AND 3
C-- STEP IS STEP SIZE FOR FIELD LINE TRACING
C-- STEQ IS STEP SIZE FOR INTEGRATION
C
      DATA RMIN,RMAX    /0.05,1.01/
      DATA STEP,STEQ    /0.20,0.03/
        BEQU=1.E10
C*****ENTRY POINT  SHELLG  TO BE USED WITH GEODETIC CO-ORDINATES       SHEL0080
      RLAT=GLAT*UMR
      CT=SIN(RLAT)                                                      SHEL0100
      ST=COS(RLAT)                                                      SHEL0110
      D=SQRT(AQUAD-(AQUAD-BQUAD)*CT*CT)
      X(1)=(ALT+AQUAD/D)*ST/ERA
      X(3)=(ALT+BQUAD/D)*CT/ERA
      RLON=GLON*UMR
      X(2)=X(1)*SIN(RLON)                                               SHEL0160
      X(1)=X(1)*COS(RLON)                                               SHEL0170
      GOTO9                                                             SHEL0180
      ENTRY SHELLC(V,FL,B0)                                             SHEL0190
C*****ENTRY POINT  SHELLC  TO BE USED WITH CARTESIAN CO-ORDINATES      SHEL0200
      X(1)=V(1)                                                         SHEL0210
      X(2)=V(2)                                                         SHEL0220
      X(3)=V(3)                                                         SHEL0230
C*****CONVERT TO DIPOL-ORIENTED CO-ORDINATES                            SHEL0240
```

```
      DATA U/                 +0.3511737,-0.9148385,-0.1993679,       SHEL0250
     A                        +0.9335804,+0.3583680,+0.0000000,       SHEL0260
     B                        +0.0714471,-0.1861260,+0.9799247/       SHEL0270
9     RQ=1./(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))
      R3H=SQRT(RQ*SQRT(RQ))                                           SHEL0290
      P(1,2)=(X(1)*U(1,1)+X(2)*U(2,1)+X(3)*U(3,1))*R3H                SHEL0300
      P(2,2)=(X(1)*U(1,2)+X(2)*U(2,2)             )*R3H               SHEL0310
      P(3,2)=(X(1)*U(1,3)+X(2)*U(2,3)+X(3)*U(3,3))*RQ                 SHEL0320
C*****FIRST THREE POINTS OF FIELD LINE                                SHEL0330
      STEP=-SIGN(STEP,P(3,2))                                         SHEL0340
      CALL STOER(P(1,2),BQ2,R2)                                       SHEL0350
      B0=SQRT(BQ2)                                                    SHEL0360
      P(1,3)=P(1,2)+0.5*STEP*P(4,2)                                   SHEL0370
      P(2,3)=P(2,2)+0.5*STEP*P(5,2)                                   SHEL0380
      P(3,3)=P(3,2)+0.5*STEP                                          SHEL0390
      CALL STOER(P(1,3),BQ3,R3)                                       SHEL0400
      P(1,1)=P(1,2)-STEP*(2.*P(4,2)-P(4,3))                           SHEL0410
      P(2,1)=P(2,2)-STEP*(2.*P(5,2)-P(5,3))                           SHEL0420
      P(3,1)=P(3,2)-STEP                                              SHEL0430
      CALL STOER(P(1,1),BQ1,R1)                                       SHEL0440
      P(1,3)=P(1,2)+STEP*(20.*P(4,3)-3.*P(4,2)+P(4,1))/18.            SHEL0450
      P(2,3)=P(2,2)+STEP*(20.*P(5,3)-3.*P(5,2)+P(5,1))/18.            SHEL0460
      P(3,3)=P(3,2)+STEP                                              SHEL0470
      CALL STOER(P(1,3),BQ3,R3)                                       SHEL0480
C*****INVERT SENSE IF REQUIRED                                        SHEL0490
      IF(BQ3.LE.BQ1)GOTO2                                             SHEL0500
      STEP=-STEP                                                      SHEL0510
      R3=R1                                                           SHEL0520
      BQ3=BQ1                                                         SHEL0530
      DO 1 I=1,7                                                      SHEL0540
      ZZ=P(I,1)                                                       SHEL0550
      P(I,1)=P(I,3)                                                   SHEL0560
1     P(I,3)=ZZ                                                       SHEL0570
C*****SEARCH FOR LOWEST MAGNETIC FIELD STRENGTH
2     IF(BQ1.LT.BEQU) THEN
        BEQU=BQ1
        IEQU=1
        ENDIF
      IF(BQ2.LT.BEQU) THEN
        BEQU=BQ2
        IEQU=2
        ENDIF
      IF(BQ3.LT.BEQU) THEN
        BEQU=BQ3
        IEQU=3
        ENDIF
C*****INITIALIZATION OF INTEGRATION LOOPS                             SHEL0580
      STEP12=STEP/12.
      STEP2=STEP+STEP                                                 SHEL0600
      STEQ=SIGN(STEQ,STEP)                                            SHEL0610
      FI=0.                                                           SHEL0620
      ICODE=1                                                         SHEL0630
      ORADIK=0.                                                       SHEL0640
      OTERM=0.                                                        SHEL0650
      STP=R2*STEQ                                                     SHEL0660
      Z=P(3,2)+STP                                                    SHEL0670
      STP=STP/0.75
      P(8,1)=STEP2*(P(1,1)*P(4,1)+P(2,1)*P(5,1))                      SHEL0690
      P(8,2)=STEP2*(P(1,2)*P(4,2)+P(2,2)*P(5,2))                      SHEL0700
C*****MAIN LOOP (FIELD LINE TRACING)                                  SHEL0710
```

```
      DO 3 N=3,3333                                                     SHEL0720
C*****CORRECTOR (FIELD LINE TRACING)                                    SHEL0730
      P(1,N)=P(1,N-1)+STEP12*(5.*P(4,N)+8.*P(4,N-1)-P(4,N-2))           SHEL0740
      P(2,N)=P(2,N-1)+STEP12*(5.*P(5,N)+8.*P(5,N-1)-P(5,N-2))           SHEL0750
C*****PREPARE EXPANSION COEFFICIENTS FOR INTERPOLATION                  SHEL0760
C*****OF SLOWLY VARYING QUANTITIES                                      SHEL0770
      P(8,N)=STEP2*(P(1,N)*P(4,N)+P(2,N)*P(5,N))                        SHEL0780
      C0=P(1,N-1)**2+P(2,N-1)**2                                        SHEL0790
      C1=P(8,N-1)                                                       SHEL0800
      C2=(P(8,N)-P(8,N-2))*0.25                                         SHEL0810
      C3=(P(8,N)+P(8,N-2)-C1-C1)/6.0
      D0=P(6,N-1)                                                       SHEL0830
      D1=(P(6,N)-P(6,N-2))*0.5                                          SHEL0840
      D2=(P(6,N)+P(6,N-2)-D0-D0)*0.5                                    SHEL0850
      E0=P(7,N-1)
      E1=(P(7,N)-P(7,N-2))*0.5                                          SHEL0870
      E2=(P(7,N)+P(7,N-2)-E0-E0)*0.5                                    SHEL0880
C*****INNER LOOP (FOR QUADRATURE)                                       SHEL0890
4     T=(Z-P(3,N-1))/STEP                                               SHEL0900
      IF(T.GT.1.)GOTO5                                                  SHEL0910
      HLI=0.5*(((C3*T+C2)*T+C1)*T+C0)                                   SHEL0920
      ZQ=Z*Z
      R=HLI+SQRT(HLI*HLI+ZQ)
      IF(R.LE.RMIN)GOTO30                                               SHEL0950
      RQ=R*R
      FF=SQRT(1.+3.*ZQ/RQ)                                              SHEL0970
      RADIK=B0-((D2*T+D1)*T+D0)*R*RQ*FF                                 SHEL0980
      IF(R-RMAX)44,44,45                                                SHEL0990
45    ICODE=2                                                           SHEL1000
      RADIK=RADIK-12.*(R-RMAX)**2                                       SHEL1010
44    IF(RADIK+RADIK.LE.ORADIK) GOTO 10
      TERM=SQRT(RADIK)*FF*((E2*T+E1)*T+E0)/(RQ+ZQ)                      SHEL1030
      FI=FI+STP*(OTERM+TERM)                                            SHEL1040
      ORADIK=RADIK                                                      SHEL1050
      OTERM=TERM                                                        SHEL1060
      STP=R*STEQ                                                        SHEL1070
      Z=Z+STP                                                           SHEL1080
      GOTO4                                                             SHEL1090
C*****PREDICTOR (FIELD LINE TRACING)                                    SHEL1100
5     P(1,N+1)=P(1,N)+STEP12*(23.*P(4,N)-16.*P(4,N-1)+5.*P(4,N-2))      SHEL1110
      P(2,N+1)=P(2,N)+STEP12*(23.*P(5,N)-16.*P(5,N-1)+5.*P(5,N-2))      SHEL1120
      P(3,N+1)=P(3,N)+STEP                                              SHEL1130
      CALL STOER(P(1,N+1),BQ3,R3)                                       SHEL1140
C*****SEARCH FOR LOWEST MAGNETIC FIELD STRENGTH
      IF(BQ3.LT.BEQU) THEN
        IEQU=N+1
        BEQU=BQ3
        ENDIF
3     CONTINUE
10    IF(IEQU.lt.2) IEQU=2
      SP(1)=P(1,IEQU-1)
      SP(2)=P(2,IEQU-1)
      SP(3)=P(3,IEQU-1)
      IF(ORADIK.LT.1E-15)GOTO11                                         SHEL1150
      FI=FI+STP/0.75*OTERM*ORADIK/(ORADIK-RADIK)
C
C-- The minimal allowable value of FI was changed from 1E-15 to 1E-12,
C-- because 1E-38 is the minimal allowable arg. for ALOG in our envir.
C-- D. Bilitza, Nov 87.
C
```

```
11     FI=0.5*ABS(FI)/SQRT(B0)+1E-12
C*****COMPUTE L FROM B AND I.  SAME AS CARMEL IN INVAR.
C
C-- Correct dipole moment is used here. D. Bilitza, Nov 87.
C
      DIMOB0=DIMO/B0
      XX=ALOG(FI*FI*FI/DIMOB0+1E-12)  !added 1E-12, 5-14-96, PRB.
      IF(XX.GT.23.0) GOTO 776
      IF(XX.GT.11.7) GOTO 775
      IF(XX.GT.+3.0) GOTO 774
      IF(XX.GT.-3.0) GOTO 773
      IF(XX.GT.-22.) GOTO 772
  771 GG=3.33338E-1*XX+3.0062102E-1
      GOTO777
  772 GG=((((((((-8.1537735E-14*XX+8.3232531E-13)*XX+1.0066362E-9)*XX+
     18.1048663E-8)*XX+3.2916354E-6)*XX+8.2711096E-5)*XX+1.3714667E-3)*
     2XX+1.5017245E-2)*XX+4.3432642E-1)*XX+6.2337691E-1
      GOTO777
  773 GG=((((((((2.6047023E-10*XX+2.3028767E-9)*XX-2.1997983E-8)*XX-
     15.3977642E-7)*XX-3.3408822E-6)*XX+3.8379917E-5)*XX+1.1784234E-3)*
     2XX+1.4492441E-2)*XX+4.3352788E-1)*XX+6.228644E-1
      GOTO777
  774 GG=((((((((6.3271665E-10*XX-3.958306E-8)*XX+9.9766148E-07)*XX-
     11.2531932E-5)*XX+7.9451313E-5)*XX-3.2077032E-4)*XX+2.1680398E-3)*
     2XX+1.2817956E-2)*XX+4.3510529E-1)*XX+6.222355E-1
      GOTO777
  775 GG=(((((2.8212095E-8*XX-3.8049276E-6)*XX+2.170224E-4)*XX-6.7310339
     1E-3)*XX+1.2038224E-1)*XX-1.8461796E-1)*XX+2.0007187E0
      GOTO777
  776 GG=XX-3.0460681E0
  777 FL=EXP(ALOG((1.+EXP(GG))*DIMOB0)/3.0)
      RETURN
C*****APPROXIMATION FOR HIGH VALUES OF L.
30    ICODE=3
      T=-P(3,N-1)/STEP
      FL=1./(ABS(((C3*T+C2)*T+C1)*T+C0)+1E-15)
      RETURN
      END
C
C
      SUBROUTINE STOER(P,BQ,R)
C*******************************************************************
C* SUBROUTINE USED FOR FIELD LINE TRACING IN SHELLG                *
C* CALLS ENTRY POINT FELDI IN GEOMAGNETIC FIELD SUBROUTINE FELDG   *
C*******************************************************************
      DIMENSION        P(7),U(3,3)
C     The following was an unlabeled common, which I have appropriately
C     named.  AJT 12-30-92.
      COMMON/BASTARDS/           XI(3),H(144)
C*****XM,YM,ZM  ARE GEOMAGNETIC CARTESIAN INVERSE CO-ORDINATES
      ZM=P(3)
      FLI=P(1)*P(1)+P(2)*P(2)+1E-15
      R=0.5*(FLI+SQRT(FLI*FLI+(ZM+ZM)**2))
      RQ=R*R
      WR=SQRT(R)
      XM=P(1)*WR
      YM=P(2)*WR
C*****TRANSFORM TO GEOGRAPHIC CO-ORDINATE SYSTEM
      DATA U/                   +0.3511737,-0.9148385,-0.1993679,
     A                          +0.9335804,+0.3583680,+0.0000000,
```

```
     B                         +0.0714471,-0.1861260,+0.9799247/          SHEL1650
      XI(1)=XM*U(1,1)+YM*U(1,2)+ZM*U(1,3)                                 SHEL1660
      XI(2)=XM*U(2,1)+YM*U(2,2)+ZM*U(2,3)                                 SHEL1670
      XI(3)=XM*U(3,1)           +ZM*U(3,3)                                SHEL1680
C*****COMPUTE DERIVATIVES                                                  SHEL1690
c      CALL FELDI(XI,H)                                                    SHEL1700
      CALL FELDI                                                          SHEL1700
      Q=H(1)/RQ                                                           SHEL1710
      DX=H(3)+H(3)+Q*XI(1)                                                SHEL1720
      DY=H(4)+H(4)+Q*XI(2)                                                SHEL1730
      DZ=H(2)+H(2)+Q*XI(3)                                                SHEL1740
C*****TRANSFORM BACK TO GEOMAGNETIC CO-ORDINATE SYSTEM                     SHEL1750
      DXM=U(1,1)*DX+U(2,1)*DY+U(3,1)*DZ                                   SHEL1760
      DYM=U(1,2)*DX+U(2,2)*DY                                             SHEL1770
      DZM=U(1,3)*DX+U(2,3)*DY+U(3,3)*DZ                                   SHEL1780
      DR=(XM*DXM+YM*DYM+ZM*DZM)/R                                         SHEL1790
C*****FORM SLOWLY VARYING EXPRESSIONS                                      SHEL1800
      P(4)=(WR*DXM-0.5*P(1)*DR)/(R*DZM)                                   SHEL1810
      P(5)=(WR*DYM-0.5*P(2)*DR)/(R*DZM)                                   SHEL1820
      DSQ=RQ*(DXM*DXM+DYM*DYM+DZM*DZM)
      BQ=DSQ*RQ*RQ
      P(6)=SQRT(DSQ/(RQ+3.*ZM*ZM))                                        SHEL1850
      P(7)=P(6)*(RQ+ZM*ZM)/(RQ*DZM)                                       SHEL1860
      RETURN                                                              SHEL1870
      END                                                                 SHEL1880
C
C
      SUBROUTINE FELDG(GLAT,GLON,ALT,BNORTH,BEAST,BDOWN,BABS)             SHEL1890
C-------------------------------------------------------------------
C CALCULATES EARTH MAGNETIC FIELD FROM SPHERICAL HARMONICS MODEL
C REF: G. KLUGE, EUROPEAN SPACE OPERATIONS CENTRE, INTERNAL NOTE 61,
C      1970.
C--------------------------------------------------------------------
C CHANGES (D. BILITZA, NOV 87):
C   - FIELD COEFFICIENTS IN BINARY DATA FILES INSTEAD OF BLOCK DATA
C   - CALCULATES DIPOL MOMENT
C--------------------------------------------------------------------
C  INPUT:  ENTRY POINT FELDG
C               GLAT  GEODETIC LATITUDE IN DEGREES (NORTH)
C               GLON  GEODETIC LONGITUDE IN DEGREES (EAST)
C               ALT   ALTITUDE IN KM ABOVE SEA LEVEL
C
C          ENTRY POINT FELDC
C               V(3)  CARTESIAN COORDINATES IN EARTH RADII (6371.2 KM)
C                       X-AXIS POINTING TO EQUATOR AT 0 LONGITUDE
C                       Y-AXIS POINTING TO EQUATOR AT 90 LONG.
C                       Z-AXIS POINTING TO NORTH POLE
C
C          COMMON BLANK AND ENTRY POINT FELDI ARE NEEDED WHEN USED
C            IN CONNECTION WITH L-CALCULATION PROGRAM SHELLG.
C
C          COMMON /MODEL/ AND /GENER/
C               UMR     = ATAN(1.0)*4./180.   <DEGREE>*UMR=<RADIANT>
C               ERA     EARTH RADIUS FOR NORMALIZATION OF CARTESIAN
C                       COORDINATES (6371.2 KM)
C               AQUAD, BQUAD   SQUARE OF MAJOR AND MINOR HALF AXIS FOR
C                       EARTH ELLIPSOID AS RECOMMENDED BY INTERNATIONAL
C                       ASTRONOMICAL UNION (6378.160, 6356.775 KM).
C               NMAX    MAXIMUM ORDER OF SPHERICAL HARMONICS
C               TIME    YEAR (DECIMAL: 1973.5) FOR WHICH MAGNETIC
```

```
C                        FIELD IS TO BE CALCULATED
C                G(M)    NORMALIZED FIELD COEFFICIENTS (SEE FELDCOF)
C                        M=NMAX*(NMAX+2)
C-----------------------------------------------------------------------
C  OUTPUT: BABS   MAGNETIC FIELD STRENGTH IN GAUSS
C          BNORTH, BEAST, BDOWN   COMPONENTS OF THE FIELD WITH RESPECT
C                 TO THE LOCAL GEODETIC COORDINATE SYSTEM, WITH AXIS
C                 POINTING IN THE TANGENTIAL PLANE TO THE NORTH, EAST
C                 AND DOWNWARD.
C-----------------------------------------------------------------------
      DIMENSION         V(3),B(3)
      CHARACTER*30      NAME    !5-14-96, change from 17 to 30
C     The following was an unlabeled common, which I have appropriately
C     named.  AJT 12-30-92.
      COMMON/BASTARDS/          XI(3),H(144)
      COMMON/MODEL/     NAME,NMAX,TIME,G(144)
      COMMON/GENER/     UMR,ERA,AQUAD,BQUAD
C
C-- IS RECORDS ENTRY POINT
C
C*****ENTRY POINT  FELDG  TO BE USED WITH GEODETIC CO-ORDINATES       SHEL1920
      IS=1                                                            SHEL1930
      RLAT=GLAT*UMR
      CT=SIN(RLAT)                                                    SHEL1950
      ST=COS(RLAT)                                                    SHEL1960
      D=SQRT(AQUAD-(AQUAD-BQUAD)*CT*CT)                               SHEL1970
      RLON=GLON*UMR
      CP=COS(RLON)                                                    SHEL1990
      SP=SIN(RLON)                                                    SHEL2000
       ZZZ=(ALT+BQUAD/D)*CT/ERA
       RHO=(ALT+AQUAD/D)*ST/ERA
       XXX=RHO*CP                                                     SHEL2030
       YYY=RHO*SP                                                     SHEL2040
      GOTO10                                                          SHEL2050
      ENTRY FELDC(V,B)                                                SHEL2060
C*****ENTRY POINT  FELDC  TO BE USED WITH CARTESIAN CO-ORDINATES      SHEL2070
      IS=2                                                            SHEL2090
      XXX=V(1)                                                        SHEL2100
      YYY=V(2)                                                        SHEL2110
      ZZZ=V(3)                                                        SHEL2120
10    RQ=1./(XXX*XXX+YYY*YYY+ZZZ*ZZZ)
      XI(1)=XXX*RQ                                                    SHEL2140
      XI(2)=YYY*RQ                                                    SHEL2150
      XI(3)=ZZZ*RQ                                                    SHEL2160
      GOTO20                                                          SHEL2170
      ENTRY FELDI                                                     SHEL2180
C*****ENTRY POINT  FELDI  USED FOR L COMPUTATION                      SHEL2190
      IS=3                                                            SHEL2200
20    IHMAX=NMAX*NMAX+1                                               SHEL2210
      LAST=IHMAX+NMAX+NMAX                                            SHEL2220
      IMAX=NMAX+NMAX-1                                                SHEL2230
      DO 8 I=IHMAX,LAST                                               SHEL2240
8     H(I)=G(I)                                                       SHEL2250
      DO 6 K=1,3,2                                                    SHEL2260
      I=IMAX                                                          SHEL2270
      IH=IHMAX                                                        SHEL2280
1     IL=IH-I                                                         SHEL2290
      F=2./FLOAT(I-K+2)                                               SHEL2300
      X=XI(1)*F                                                       SHEL2310
      Y=XI(2)*F                                                       SHEL2320
```

```
      Z=XI(3)*(F+F)                                                     SHEL2330
      I=I-2                                                             SHEL2340
      IF(I-1)5,4,2                                                      SHEL2350
2     DO 3 M=3,I,2                                                      SHEL2360
      H(IL+M+1)=G(IL+M+1)+Z*H(IH+M+1)+X*(H(IH+M+3)-H(IH+M-1))           SHEL2370
     A                                -Y*(H(IH+M+2)+H(IH+M-2))          SHEL2380
3     H(IL+M)=G(IL+M)+Z*H(IH+M)+X*(H(IH+M+2)-H(IH+M-2))                 SHEL2390
     A                          +Y*(H(IH+M+3)+H(IH+M-1))                SHEL2400
4     H(IL+2)=G(IL+2)+Z*H(IH+2)+X*H(IH+4)-Y*(H(IH+3)+H(IH))             SHEL2410
      H(IL+1)=G(IL+1)+Z*H(IH+1)+Y*H(IH+4)+X*(H(IH+3)-H(IH))             SHEL2420
5     H(IL)=G(IL)+Z*H(IH)+2.*(X*H(IH+1)+Y*H(IH+2))                      SHEL2430
      IH=IL                                                             SHEL2440
      IF(I.GE.K)GOTO1                                                   SHEL2450
6     CONTINUE                                                          SHEL2460
      IF(IS.EQ.3)RETURN                                                 SHEL2470
      S=.5*H(1)+2.*(H(2)*XI(3)+H(3)*XI(1)+H(4)*XI(2))                   SHEL2480
      T=(RQ+RQ)*SQRT(RQ)                                                SHEL2490
      BXXX=T*(H(3)-S*XXX)                                               SHEL2500
      BYYY=T*(H(4)-S*YYY)                                               SHEL2510
      BZZZ=T*(H(2)-S*ZZZ)                                               SHEL2520
      IF(IS.EQ.2)GOTO7                                                  SHEL2530
      BABS=SQRT(BXXX*BXXX+BYYY*BYYY+BZZZ*BZZZ)
      BEAST=BYYY*CP-BXXX*SP                                             SHEL2550
      BRHO=BYYY*SP+BXXX*CP                                              SHEL2560
      BNORTH=BZZZ*ST-BRHO*CT                                            SHEL2570
      BDOWN=-BZZZ*CT-BRHO*ST                                            SHEL2580
      RETURN                                                            SHEL2590
7     B(1)=BXXX                                                         SHEL2600
      B(2)=BYYY                                                         SHEL2610
      B(3)=BZZZ                                                         SHEL2620
      RETURN                                                            SHEL2630
      END                                                               SHEL2640
C
C
        SUBROUTINE FELDCOF(YEAR,DIMO)
C-----------------------------------------------------------------------
C  DETERMINES COEFFICIENTS AND DIPOL MOMENT FROM IGRF MODELS
C
C       INPUT:  YEAR    DECIMAL YEAR FOR WHICH GEOMAGNETIC FIELD IS TO
C                       BE CALCULATED
C       OUTPUT: DIMO    GEOMAGNETIC DIPOL MOMENT IN GAUSS (NORMALIZED
C                       TO EARTH'S RADIUS) AT THE TIME (YEAR)
C  D. BILITZA, NSSDC, GSFC, CODE 633, GREENBELT, MD 20771,
C       (301)286-9536   NOV 1987.
C
C  Modified by AJT 12-9-92:
C  allow for multiple calls: field paramters are not read in unless
C  year value is changed.  Call to INITIZE also added here.
C-----------------------------------------------------------------------
        CHARACTER*30    FILMOD, FIL1, FIL2   !5-14-96, change from 17 to 30
        DIMENSION       GH1(144),GH2(120),GHA(144),FILMOD(11),DTEMOD(11)
        DOUBLE PRECISION X,F0,F
        COMMON/MODEL/   FIL1,NMAX,TIME,GH1
        COMMON/GENER/   UMR,ERAD,AQUAD,BQUAD

        DATA            FILMOD/
C     &                  'creme96:dgrf45.dat',
C     &                  'creme96:dgrf50.dat',
C     1                 'creme96:dgrf55.dat', 'creme96:dgrf60.dat',
C     &                  'creme96:dgrf65.dat',
```

```
C     2                    'creme96:dgrf70.dat', 'creme96:dgrf75.dat',
C     &                     'creme96:dgrf80.dat',
C     3                    'creme96:dgrf85.dat', 'creme96:igrf90.dat',
C     &                     'creme96:igrf90s.dat'/
C       Remove directory path, per BB's new file open routines AJT 11/18/97
     &                     'dgrf45.dat',
     &                     'dgrf50.dat',
     1                     'dgrf55.dat', 'dgrf60.dat',
     &                     'dgrf65.dat',
     2                     'dgrf70.dat', 'dgrf75.dat',
     &                     'dgrf80.dat',
     3                     'dgrf85.dat', 'igrf90.dat',
     &                     'igrf90s.dat'/
      DATA                 DTEMOD / 1945., 1950., 1955., 1960., 1965.,
     1                     1970., 1975., 1980., 1985., 1990., 1995./
      DATA YEAROLD/0/,IENT/0/
      COMMON/AJTDIMO/AJTDIMO

      IF (IENT.EQ.0) THEN
          IENT=1
          write(*,*)' Initialization call to FELDCOF: YEAR = ',YEAR
          CALL INITIZE
      ENDIF

      IF (ABS(YEAR-YEAROLD).LT.0.001) THEN
          DIMO=AJTDIMO
          RETURN
      ENDIF
      YEAROLD=YEAR
c
c  numye is number of years represented by IGRF models
c
      NUMYE=10
C
C  IS=0 FOR SCHMIDT NORMALIZATION   IS=1 GAUSS NORMALIZATION
C  IU  IS INPUT UNIT NUMBER FOR IGRF COEFFICIENT SETS
C
      IU = 10
      IS = 0
C-- DETERMINE IGRF-YEARS FOR INPUT-YEAR
      TIME = YEAR
      IYEA = INT(YEAR/5.)*5
      L = (IYEA - 1945)/5 + 1
      IF(L.LT.1) L=1
      IF(L.GT.NUMYE) L=NUMYE
      DTE1 = DTEMOD(L)
      FIL1 = FILMOD(L)
      DTE2 = DTEMOD(L+1)
      FIL2 = FILMOD(L+1)
C-- GET IGRF COEFFICIENTS FOR THE BOUNDARY YEARS
C   Error messages added by AJT 11/24/97
      CALL GETSHC (IU, FIL1, NMAX1, ERAD, GH1, IER)
          IF (IER .NE. 0) THEN
              WRITE(6,9999) FIL1,IER
 9999         FORMAT('@ 02001 ABNORMAL TERMINATION: ',
     &            /,1x,' IGRF Coefficient file not found: ',
     &            /,1x,A80,
     &            /,1x,' Error return code = ',I10,' STOP.')
              STOP
          ENDIF
```

```
        CALL GETSHC (IU, FIL2, NMAX2, ERAD, GH2, IER)
            IF (IER .NE. 0) THEN
                WRITE(6,9999) FIL2,IER
                STOP
            ENDIF
C-- DETERMINE IGRF COEFFICIENTS FOR YEAR
        IF (L .LE. NUMYE-1) THEN
          CALL INTERSHC (YEAR, DTE1, NMAX1, GH1, DTE2,
     1          NMAX2, GH2, NMAX, GHA)
        ELSE
          CALL EXTRASHC (YEAR, DTE1, NMAX1, GH1, NMAX2,
     1          GH2, NMAX, GHA)
        ENDIF
C-- DETERMINE MAGNETIC DIPOL MOMENT AND COEFFIECIENTS G
        F0=0.D0
        DO 1234 J=1,3
           F = GHA(J) * 1.D-5
           F0 = F0 + F * F
1234    CONTINUE
        DIMO = DSQRT(F0)
        AJTDIMO=DIMO

        GH1(1) =  0.0
        I=2
        F0=1.D-5
        IF(IS.EQ.0) F0=-F0
        SQRT2=SQRT(2.)

      DO 9 N=1,NMAX
        X = N
        F0 = F0 * X * X / (4.D0 * X - 2.D0)
        IF(IS.EQ.0) F0 = F0 * (2.D0 * X - 1.D0) / X
        F = F0 * 0.5D0
        IF(IS.EQ.0) F = F * SQRT2
        GH1(I) = GHA(I-1) * F0
        I = I+1
      DO 9 M=1,N
        F = F * (X + M) / (X - M + 1.D0)
        IF(IS.EQ.0) F = F * DSQRT((X - M + 1.D0) / (X + M))
        GH1(I) = GHA(I-1) * F
        GH1(I+1) = GHA(I) * F
        I=I+2
9     CONTINUE
        RETURN
        END
C
C
        SUBROUTINE GETSHC (IU, FSPEC, NMAX, ERAD, GH, IER)

C ===============================================================
C
C       Version 1.01
C
C       Reads spherical harmonic coefficients from the specified
C       file into an array.
C
C       Input:
C           IU    - Logical unit number
C           FSPEC - File specification
C
```

```
C       Output:
C           NMAX  - Maximum degree and order of model
C           ERAD  - Earth's radius associated with the spherical
C                   harmonic coefficients, in the same units as
C                   elevation
C           GH    - Schmidt quasi-normal internal spherical
C                   harmonic coefficients
C           IER   - Error number: =  0, no error
C                                 = -2, records out of order
C                                 = FORTRAN run-time error number
C
C       A. Zunde
C       USGS, MS 964, Box 25046 Federal Center, Denver, CO  80225
C
C ===============================================================

        CHARACTER       FSPEC*(*)
        DIMENSION       GH(*)
        integer         stat,creme96_open

C ---------------------------------------------------------------
C       Open coefficient file. Read past first header record.
C       Read degree and order of model and Earth's radius.
C ---------------------------------------------------------------

C       made READONLY, 5-16-96, PRB.
c       OPEN (IU,FILE=FSPEC,STATUS='OLD',READONLY,IOSTAT=IER,ERR=999)
        stat = creme96_open(fspec,'cr96tables',iu,'old')
        if (stat .ne. 0) goto 999
        READ (IU, *, IOSTAT=IER, ERR=999)
        READ (IU, *, IOSTAT=IER, ERR=999) NMAX, ERAD

C ---------------------------------------------------------------
C       Read the coefficient file, arranged as follows:
C
C                                         N     M     G     H
C
C                                         ---------------------
C                                     /   1     0    GH(1)  -
C                                    /    1     1    GH(2) GH(3)
C                                   /     2     0    GH(4)  -
C                                  /      2     1    GH(5) GH(6)
C           NMAX*(NMAX+3)/2       /       2     2    GH(7) GH(8)
C              records            \       3     0    GH(9)  -
C                                  \      .     .     .     .
C                                   \     .     .     .     .
C           NMAX*(NMAX+2)            \    .     .     .     .
C           elements in GH            \ NMAX  NMAX    .     .
C
C       N and M are, respectively, the degree and order of the
C       coefficient.
C ---------------------------------------------------------------

        I = 0
        DO 2211 NN = 1, NMAX
            DO 2233 MM = 0, NN
                READ (IU, *, IOSTAT=IER, ERR=999) N, M, G, H
                IF (NN .NE. N .OR. MM .NE. M) THEN
                    IER = -2
                    GOTO 999
                ENDIF
```

```
                I = I + 1
                GH(I) = G
                IF (M .NE. 0) THEN
                    I = I + 1
                    GH(I) = H
                ENDIF
2233        CONTINUE
2211    CONTINUE

999     CLOSE (IU)

        RETURN
        END
C
C
        SUBROUTINE INTERSHC (DATE, DTE1, NMAX1, GH1, DTE2,
     1                       NMAX2, GH2, NMAX, GH)

C ===============================================================
C
C       Version 1.01
C
C       Interpolates linearly, in time, between two spherical
C       harmonic models.
C
C       Input:
C           DATE  - Date of resulting model (in decimal year)
C           DTE1  - Date of earlier model
C           NMAX1 - Maximum degree and order of earlier model
C           GH1   - Schmidt quasi-normal internal spherical
C                   harmonic coefficients of earlier model
C           DTE2  - Date of later model
C           NMAX2 - Maximum degree and order of later model
C           GH2   - Schmidt quasi-normal internal spherical
C                   harmonic coefficients of later model
C
C       Output:
C           GH    - Coefficients of resulting model
C           NMAX  - Maximum degree and order of resulting model
C
C       A. Zunde
C       USGS, MS 964, Box 25046 Federal Center, Denver, CO  80225
C
C ===============================================================

        DIMENSION       GH1(*), GH2(*), GH(*)

C ---------------------------------------------------------------
C       The coefficients (GH) of the resulting model, at date
C       DATE, are computed by linearly interpolating between the
C       coefficients of the earlier model (GH1), at date DTE1,
C       and those of the later model (GH2), at date DTE2. If one
C       model is smaller than the other, the interpolation is
C       performed with the missing coefficients assumed to be 0.
C ---------------------------------------------------------------

        FACTOR = (DATE - DTE1) / (DTE2 - DTE1)

        IF (NMAX1 .EQ. NMAX2) THEN
            K = NMAX1 * (NMAX1 + 2)
```

```
            NMAX = NMAX1
        ELSE IF (NMAX1 .GT. NMAX2) THEN
            K = NMAX2 * (NMAX2 + 2)
            L = NMAX1 * (NMAX1 + 2)
            DO 1122 I = K + 1, L
1122            GH(I) = GH1(I) + FACTOR * (-GH1(I))
            NMAX = NMAX1
        ELSE
            K = NMAX1 * (NMAX1 + 2)
            L = NMAX2 * (NMAX2 + 2)
            DO 1133 I = K + 1, L
1133            GH(I) = FACTOR * GH2(I)
            NMAX = NMAX2
        ENDIF

        DO 1144 I = 1, K
1144        GH(I) = GH1(I) + FACTOR * (GH2(I) - GH1(I))

        RETURN
        END
C
C
        SUBROUTINE EXTRASHC (DATE, DTE1, NMAX1, GH1, NMAX2,
     1                       GH2, NMAX, GH)

C ================================================================
C
C       Version 1.01
C
C       Extrapolates linearly a spherical harmonic model with a
C       rate-of-change model.
C
C       Input:
C           DATE  - Date of resulting model (in decimal year)
C           DTE1  - Date of base model
C           NMAX1 - Maximum degree and order of base model
C           GH1   - Schmidt quasi-normal internal spherical
C                   harmonic coefficients of base model
C           NMAX2 - Maximum degree and order of rate-of-change
C                   model
C           GH2   - Schmidt quasi-normal internal spherical
C                   harmonic coefficients of rate-of-change model
C
C       Output:
C           GH    - Coefficients of resulting model
C           NMAX  - Maximum degree and order of resulting model
C
C       A. Zunde
C       USGS, MS 964, Box 25046 Federal Center, Denver, CO  80225
C
C ================================================================

        DIMENSION       GH1(*), GH2(*), GH(*)

C ----------------------------------------------------------------
C       The coefficients (GH) of the resulting model, at date
C       DATE, are computed by linearly extrapolating the coef-
C       ficients of the base model (GH1), at date DTE1, using
C       those of the rate-of-change model (GH2), at date DTE2. If
C       one model is smaller than the other, the extrapolation is
```

```
C       performed with the missing coefficients assumed to be 0.
C ----------------------------------------------------------------

        FACTOR = (DATE - DTE1)

        IF (NMAX1 .EQ. NMAX2) THEN
            K = NMAX1 * (NMAX1 + 2)
            NMAX = NMAX1
        ELSE IF (NMAX1 .GT. NMAX2) THEN
            K = NMAX2 * (NMAX2 + 2)
            L = NMAX1 * (NMAX1 + 2)
            DO 1155 I = K + 1, L
1155            GH(I) = GH1(I)
            NMAX = NMAX1
        ELSE
            K = NMAX1 * (NMAX1 + 2)
            L = NMAX2 * (NMAX2 + 2)
            DO 1166 I = K + 1, L
1166            GH(I) = FACTOR * GH2(I)
            NMAX = NMAX2
        ENDIF

        DO 1177 I = 1, K
1177        GH(I) = GH1(I) + FACTOR * GH2(I)

        RETURN
        END
C
C
        SUBROUTINE INITIZE
C----------------------------------------------------------------
C Initializes the parameters in COMMON/GENER/
C
C       UMR     = ATAN(1.0)*4./180.   <DEGREE>*UMR=<RADIANT>
C       ERA     EARTH RADIUS FOR NORMALIZATION OF CARTESIAN
C                       COORDINATES (6371.2 KM)
C       EREQU   MAJOR HALF AXIS FOR EARTH ELLIPSOID (6378.160 KM)
C       ERPOL   MINOR HALF AXIS FOR EARTH ELLIPSOID (6356.775 KM)
C       AQUAD   SQUARE OF MAJOR HALF AXIS FOR EARTH ELLIPSOID
C       BQUAD   SQUARE OF MINOR HALF AXIS FOR EARTH ELLIPSOID
C
C ERA, EREQU and ERPOL as recommended by the INTERNATIONAL
C ASTRONOMICAL UNION .
C-----------------------------------------------------------------
        COMMON/GENER/   UMR,ERA,AQUAD,BQUAD
        DATA IENT/0/
        IF (IENT.EQ.0) THEN
            write(*,*)' Initialization call to INITIZE'
            IENT=1
        ENDIF
        ERA=6371.2
        EREQU=6378.16
        ERPOL=6356.775
        AQUAD=EREQU*EREQU
        BQUAD=ERPOL*ERPOL
        UMR=ATAN(1.0)*4./180.
        RETURN
        END
```

```
      PROGRAM SHIELDFILE_DRIVER
C
C     This is an auxilliary program to the CREME96 software, which translates
C     a user-supplied shielding distribution into a file with standard
C     format and header information, as required by the CREME96 software.
C     From the user inputs, supplied via interactive dialogue, this program
C     produces an output file.  The suggested name of this output file
C     is something.SHD. (ie. the extension should be SHD.)  If the file is
C     given some other extension, it will not be accessible by standard
C     CREME96 directory features and (in the WWW version) pull-down menus.
C
      IMPLICIT NONE
      CHARACTER*80 SHIELDFILE,COMMENT
      CHARACTER*12 MATERIAL
      INTEGER*4 IUNITS,NBINS,MAXSHIELD
      PARAMETER (MAXSHIELD=500)
      REAL*4 XTHICK0(MAXSHIELD),XPROB0(MAXSHIELD)
      REAL*4 XTHICK(MAXSHIELD), XPROB(MAXSHIELD)
      REAL*4 XMEAN,XRMS,TOTAL
      INTEGER*4 ERRFLAG
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE
C
      CALL GET_CREME96_VERSION(VERSION_NUMBER)
      PROGRAM_CODE=7
C
      CALL INISHIELD(MAXSHIELD,COMMENT,
     &               IUNITS,MATERIAL,NBINS,XTHICK0,XPROB0,
     &               SHIELDFILE)

      CALL CHECK_SHIELD_DISTRIBUTION(NBINS,XTHICK0,XPROB0,
     &                               XTHICK,XPROB,
     &                               XMEAN,XRMS,TOTAL,ERRFLAG)
C
      CALL OUTPUT_SHIELDFILE(SHIELDFILE,
     *                       COMMENT,IUNITS,MATERIAL,
     *                       NBINS,XTHICK,XPROB,
     *                       XMEAN,XRMS,TOTAL,ERRFLAG,
     *                       VERSION_NUMBER,PROGRAM_CODE)
C
      STOP
      END
```

```
      REAL FUNCTION SOLAR_HEAVY_IONS(IZ,EN,IMODE0,ERRFLUX)
C
C     Returns the event-integrated interplanetary solar energetic heavy
C     ion flux (IZ > 2) for element IZ at energy E (in MeV/nuc) for
C     SEP event LSEP.
C
C     Inputs:
C        IZ     = Atomic number (IZ=3-30)
C        EN     = Energy (MeV/nuc)
C        IMODE0 = 1: 'worst day' based on measurements of 20 OCT 89 event
C        IMODE0 = 2: 'worst week' based on measurements of 19-26 OCT 89 events.
C        IMODE0 = 3: peak flux, based on 5-minute-averaged GOES protons
C                    on 20OCT89
C
C     Outputs:
C
C          SOLAR_HEAVY_IONS = event integrated flux [in (m2-sr-MeV/nuc)**-1]
C                 at energy E;  NOTE: NOT divided by TIME!
C          ERRFLUX = its error, based on error propagation of the
C                 fit parameters. Not yet available.
C

      IMPLICIT NONE
      REAL*4 EN,ERRFLUX,FLXDUM
      INTEGER*4 IZ,IMODE0,IMODE,IUSE,IFIT,MELM
      INTEGER*4 NFITS,NMODE,NZ
      PARAMETER (NFITS=2,NMODE=2,NZ=20,MELM=92)
      REAL*4 A1(NFITS,NMODE),A2(NFITS,NMODE),A3(NFITS,NMODE)
      REAL*4 EB1(NFITS,NMODE),EB2(NFITS,NMODE)
      REAL*4 BETA(NFITS,NMODE),G(NFITS,NMODE),GAMMA(NFITS,NMODE)
      REAL*4 RELNORM,AVESEP,PEAKFAC,SEP_PEAK_FACTOR
      DIMENSION RELNORM(NZ),AVESEP(MELM)
      DATA A1/  2.3759E+06, 2.4218E+5, 2.9731E+06, 3.2764E+05 /
      DATA A2/  4.9518E+08, 1.8991E+08, 1.1307E+09, 3.0372E+08/
      DATA A3/  0.106702E+10, 0.252948E+10, 0.667628E+09, 0.249719E+09/
      DATA EB1/  4*0.0 /
      DATA EB2/  15.94, 24.23, 12.89, 17.22/
      DATA BETA/ 0.5601, 0.2967, 0.4372, 0.2507/
      DATA G/    5.7000, 5.7000, 5.7000, 5.7000/
      DATA GAMMA/4.14060, 4.52970, 3.76850, 3.7610/
      DATA RELNORM/5*0.0,0.47049,0.12059,1.00000,0.0,
     &              0.21312,0.0,0.20624,0.0,0.35935,0.0,
     &              0.09758,0.0,
     &              0.00000,0.0,0.04826/
C
C     For otherwise undetermined elements Z=6-30, use relative abundances
C     at 10 MeV/nuc as determined by Croley et al. from the Galileo
C     data for the 24OCT89 event, since these are the best available
C     observations.  AVESEP contains nominal abundances, relative to Fe:
C
      DATA AVESEP/5*0.0,
     &  0.4263E+01, 0.1567E+01, 0.1230E+02, 0.5610E-03, 0.1915E+01,
     &  0.2146E+00, 0.3650E+01, 0.2247E+00, 0.2280E+01, 0.2804E-02,
     &  0.1252E+00, 0.2067E-02, 0.2179E-01, 0.4483E-02, 0.9506E-01,
     &  0.2929E-03, 0.4377E-02, 0.4088E-03, 0.1650E-01, 0.5625E-02,
     &  0.1000E+01, 0.1303E-01, 0.3172E-01, 0.3048E-03, 0.7457E-03,
C
C     For Z>30 elements, include nominal solar abundances as in old CREME.
C
     &   0.4878E-04, 0.1220E-03, 0.7317E-05, 0.7317E-04, 0.9756E-05,
```

```
     &   0.4878E-04, 0.7317E-05, 0.2439E-04, 0.4878E-05, 0.1220E-04,
     &   0.9756E-06, 0.4878E-05, 0.0000E+00, 0.2195E-05, 0.4878E-06,
     &   0.1463E-05, 0.4878E-06, 0.1707E-05, 0.2195E-06, 0.4878E-05,
     &   0.3415E-06, 0.7317E-05, 0.1463E-05, 0.6585E-05, 0.4878E-06,
     &   0.4878E-05, 0.4878E-06, 0.1220E-05, 0.1951E-06, 0.9756E-06,
     &   0.0000E+00, 0.2439E-06, 0.9756E-07, 0.4878E-06, 0.7317E-07,
     &   0.4878E-06, 0.9756E-07, 0.2439E-06, 0.4878E-07, 0.2195E-06,
     &   0.4878E-07, 0.1951E-06, 0.2195E-07, 0.2439E-06, 0.4878E-07,
     &   0.7317E-06, 0.7317E-06, 0.1463E-05, 0.2439E-06, 0.2439E-06,
     &   0.2195E-06, 0.2439E-05, 0.1463E-06, 0.0000E+00, 0.0000E+00,
     &   0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.4878E-07,
     &   0.0000E+00, 0.2927E-07/

C
      SOLAR_HEAVY_IONS=0.0
      ERRFLUX=0.0
      IF (EN.LE.0.) RETURN
      IF (IZ.LE.2)  RETURN

C     Set mode flag:
C     Note: Peak flux (IMODE0=3) is scaled from worst-day (IMODE0=1):

      IMODE=IMODE0
      IF (IMODE0.EQ.3) IMODE=1

C     Select baseline spectrum:

      IUSE=8
      IFIT=1
      IF (IZ.GT.20) THEN
                    IUSE=26
                    IFIT=2
      ENDIF

C     Nominal modeling:

      IF (EN.LE.EB1(IFIT,IMODE)) THEN
C         Note: EB1=0: This segment never activated.
          FLXDUM=A1(IFIT,IMODE)*EXP(-BETA(IFIT,IMODE)*EN)
      ELSEIF (EB1(IFIT,IMODE).LT.EN .and. EN.LE.EB2(IFIT,IMODE)) THEN
C         Forced exponential roll-off at low energies
          FLXDUM=EXP(-G(IFIT,IMODE)*EN**0.25)
          FLXDUM=A2(IFIT,IMODE)*FLXDUM*EN**0.25
      ELSEIF (EN.GT.EB2(IFIT,IMODE)) THEN
C         Power-law fits at high energies
          FLXDUM=A3(IFIT,IMODE)*EN**(-GAMMA(IFIT,IMODE))
      ENDIF

C     Special case of broken power-law in worst-week Fe:

      IF (IMODE.EQ.2 .and. IUSE.EQ.26 .and. EN.GT. 127.93) THEN
          FLXDUM=3.168141E+6*EN**(-2.861)
      ENDIF

C     Scale by relative-abundance factors to get other elements

      IF (IZ.LE.20 .and. RELNORM(IZ).GT.0.) THEN
          SOLAR_HEAVY_IONS=FLXDUM*RELNORM(IZ)/RELNORM(IUSE)
      ELSE
          SOLAR_HEAVY_IONS=FLXDUM*AVESEP(IZ)/AVESEP(IUSE)
```

```
      ENDIF

C     Finally, convert from /cm2 to /m2:
      SOLAR_HEAVY_IONS=SOLAR_HEAVY_IONS*1.0E+4

C     For peak-flux model, scale from average:
      IF (IMODE0.EQ.3) THEN
          PEAKFAC=SEP_PEAK_FACTOR(EN)
          SOLAR_HEAVY_IONS=SOLAR_HEAVY_IONS*PEAKFAC
      ENDIF
      RETURN
      END

      REAL FUNCTION SEP_PEAK_FACTOR(EN)
C
C     Gets relative scale factor of peak-to-average flux, based on
C     GOES proton observations of the 20OCT89 SEP event:
C
      IMPLICIT NONE
      REAL EN,EN0,ERRFLUX,SOLAR_PROTONS,AVEFLUX,PEAKFLUX
      CHARACTER*7 LABEL
      EN0=EN
      IF (EN0.GT.400.) EN0=400.

      LABEL='20OCT89'
      AVEFLUX=SOLAR_PROTONS(EN0,LABEL,ERRFLUX)
      LABEL='PEAKFLX'
      PEAKFLUX=SOLAR_PROTONS(EN0,LABEL,ERRFLUX)
      SEP_PEAK_FACTOR=0.0
      IF (AVEFLUX.LE.0.0.or.PEAKFLUX.LE.0.0) RETURN
      SEP_PEAK_FACTOR=PEAKFLUX/AVEFLUX
      RETURN
      END
```

```
      REAL FUNCTION SOLAR_PROTONS(E,LSEP,ERRFLUX)
C
C     Returns the event-integrated interplanetary Solar Energetic Proton
C     fluence at energy E (in MeV) for SEP event "LSEP" and its estimated
C     error.
C
C     Fluences are based on fits to the MEPAD integral proton channels
C     on GOES-7 and HEPAD proton channels on GOES-6.
C
C     Inputs:
C       E    = Energy (MeV/nuc)
C       LSEP = SEP event label (CHAR*7, see table below)
C     Outputs:
C       SEP_PROTONS = event integrated proton flux [in (m2-sr-MeV/nuc)**-1]
C                            at energy E;  NOTE: NOT divided by TIME!
C       SIGFLUX =     its error; nominally set to 10%
C
C
C     This routine get fluence (differential in energy), starting from
C     the fits to the integral rigidity spectrum.
C
      IMPLICIT NONE
      INTEGER*4 NTERMS,NEVTS,IZ,ISEP
      PARAMETER (NTERMS=4,NEVTS=5)
      REAL*4 E,ERRFLUX,COEF,DAMU,Q,AN,RIGVAL,MAGNETIC_RIGIDITY
      REAL*4 XVAL,XDUM,GG,YVAL,FACTOR
      CHARACTER*7 LSEP
      INTEGER*4 K

      DATA DAMU/931.5016/
      DATA IZ/1/,Q/1.0/,AN/1.0/
      DIMENSION COEF(NTERMS,NEVTS)
      DATA COEF/
C 20OCT89
     &  0.231924E+02, -0.223621E+02,  0.168443E+02, -0.599497E+01,
C 19OCT89
     &  0.207522E+02, -0.139185E+02,  0.874864E+01, -0.334163E+01,
C 22OCT89
     &  0.215929E+02, -0.155329E+02,  0.790699E+01, -0.258523E+01,
C 24OCT89
     &  0.214273E+02, -0.193048E+02,  0.164647E+02, -0.632318E+01,
C Peak fluence (/cm2-sr-s) October 1989:
     &  0.135472E+02, -0.232970E+02,  0.185617E+02, -0.674944E+01/

      SOLAR_PROTONS=0.0
      ERRFLUX=0.0
      IF (E.LT.1.0 .OR. E.GT.1.0E+5) RETURN
      ISEP=0
      IF (LSEP.EQ.'20OCT89') ISEP=1
      IF (LSEP.EQ.'19OCT89') ISEP=2
      IF (LSEP.EQ.'22OCT89') ISEP=3
      IF (LSEP.EQ.'24OCT89') ISEP=4
      IF (LSEP.EQ.'PEAKFLX') ISEP=5
      IF (ISEP.EQ.0) RETURN

         RIGVAL=MAGNETIC_RIGIDITY(E,Q,AN)

         XVAL=COEF(1,ISEP)
         XDUM=0.0
         DO 500 K=2,NTERMS
```

```
          GG=FLOAT(K-1)
          XVAL=XVAL+COEF(K,ISEP)*RIGVAL**GG
          XDUM=XDUM+GG*COEF(K,ISEP)*RIGVAL**(GG-1.0)
 500    CONTINUE

        YVAL=ABS(XDUM)*EXP(XVAL)


C     Now need to calculate Jacobian to go from rigidity to kinetic
C     energy:

      FACTOR=(E+DAMU)/SQRT(E*E+2*E*DAMU)
      FACTOR=AN*FACTOR/Q
C
C     Additional factor comes from two sources:
C       1.0E-3 comes from GeV to MeV conversion;
C       1.0E+4 comes from /cm2 to /m2 conversion.
C
      SOLAR_PROTONS=10.0*FACTOR*YVAL
C
C     For the peak flux mode, the fit parameters give the 5-minute
C     averaged fluence in (cm2-sr-s)**-1.  Need to remove time factor
C     to put on same basis as other fits:
C
      IF (ISEP.EQ.5) SOLAR_PROTONS=SOLAR_PROTONS*300.0

      IF (SOLAR_PROTONS.LT.0.) SOLAR_PROTONS=0.0
      ERRFLUX=0.10*SOLAR_PROTONS
      RETURN
      END
```

```
      SUBROUTINE STABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET)
      IMPLICIT NONE
      REAL*4 ELOWER,EUPPER,AA,AMASS,EE,DE,STPOW
      INTEGER*4 M,IZLO,IZUP,NELM,MARR,STAT,CREME96_OPEN
      CHARACTER*12 TARGET
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 SP(NELM,MARR),E(MARR)
      INTEGER*4 J,K,I
      COMMON/MASS/AMASS(109)

C     Construct list of energies

        DE=(EUPPER/ELOWER)**(1./(M-1.))
        E(1)=ELOWER
        DO J=2,M-1
          E(J)=E(J-1)*DE
        END DO
        E(M)=EUPPER

c        OPEN(UNIT=28,STATUS='NEW',FILE='USER:STABLE.DAT')
        stat = creme96_open('stable.dat','user',28,'new')
        WRITE(28,100) ELOWER,EUPPER,M,IZLO,IZUP,TARGET
        WRITE(28,100)
        DO J=IZLO,IZUP
          AA=AMASS(J)
          DO K=1,M
            EE=E(K)
            SP(J,K)=STPOW(EE,FLOAT(J),AA,TARGET)*AA
          END DO
          WRITE(28,200) (SP(J,K),K=1,M)
C         Skip line between elements AJT 5/7/96
          WRITE(28,200)
        END DO
        CLOSE(UNIT=28)

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,2X,1PE10.4)
200   FORMAT((1X,6(1PE10.4,2X)))

      RETURN
      END
```

```
      FUNCTION STPOW(E1,Z0,A1,NAME)
C     ***************************************************************
C     *  THIS ROUTINE RETURNS THE STOPPING POWER OF NUCLIDE (Z0,A1)
C     *  IN MATERIAL 'NAME' AT E1 (MeV/nucleon).
C     *  DATA ON THE STOPPING MATERIAL IS CONTAINED IN TARGET.DAT.
C     ***************************************************************
      CHARACTER*12 NAME$(150),NAME,LNAME
      REAL IADJ$(150,28),NA$(150,28),NASPM$(150,28)
      INTEGER*4 STAT,CREME96_OPEN
      DIMENSION NAS$(150),NZ$(150,28),DENS$(150)
      DIMENSION IGAS$(150),ETAD$(150)
      DATA ITARG,LNAME/0,'QXZ8F3'/
      COMMON /AVG/ AVGZ,AVGZ2,AVGA,AVGI  ! MEAN STOPPING MED. PARAMETERS
C     *  READ IN TARGET DATA
      IF (ITARG.EQ.1) GO TO 100
c     OPEN(UNIT=10,FILE='CREME96:TARGET.DAT',STATUS='OLD',READONLY,SHARED)
      stat = creme96_open('target.dat','cr96tables',10,'old')
 1    FORMAT(1X,I3)
 2    FORMAT(1X,A12,2X,F9.6,2X,F9.6,2X,I1,2X,I2)
 3    FORMAT(1X,I3,2X,F8.4,2X,F5.1,2X,F9.5)
      READ(10,1) NM$
      DO J1=1,NM$
        READ(10,2) NAME$(J1),DENS$(J1),ETAD$(J1),IGAS$(J1),NAS$(J1)
        DO J2=1,NAS$(J1)
          READ(10,3) NZ$(J1,J2),NA$(J1,J2),IADJ$(J1,J2),NASPM$(J1,J2)
        END DO
      END DO
      ITARG=1
      CLOSE(UNIT=10)
 100  CONTINUE

C     *  DETERMINE WHICH TARGET DATA TO USE
      IF (NAME.EQ.LNAME) GO TO 200
      DO J1=1,NM$
        IF(NAME.EQ.NAME$(J1)) K1=J1
      END DO
      LNAME=NAME$(K1)
      IF (NAME.NE.LNAME) THEN
        STPOW=0.
        RETURN
      ENDIF

C     *  COMPUTE MATERIAL PARAMETERS
      RHO=DENS$(K1)
      IGAS=IGAS$(K1)
      ETAD=ETAD$(K1)
      NTOTAL=0
      AVGZ=0.
      AVGZ2=0.
      AVGA=0.
      AVGI=0.
      DO J1=1,NAS$(K1)
        NTOTAL=NTOTAL + NASPM$(K1,J1)
        AVGZ=AVGZ + NASPM$(K1,J1)*FLOAT(NZ$(K1,J1))
        AVGZ2=AVGZ2 + NASPM$(K1,J1)*FLOAT(NZ$(K1,J1))**2
        AVGA=AVGA + NASPM$(K1,J1)*NA$(K1,J1)
        AVGI=AVGI + NASPM$(K1,J1)*ALOG(IADJ$(K1,J1))
      END DO
      AVGZ=AVGZ/FLOAT(NTOTAL)
      AVGZ2=AVGZ2/FLOAT(NTOTAL)
```

```
      SUBROUTINE STABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET)
      IMPLICIT NONE
      REAL*4 ELOWER,EUPPER,AA,AMASS,EE,DE,STPOW
      INTEGER*4 M,IZLO,IZUP,NELM,MARR,STAT,CREME96_OPEN
      CHARACTER*12 TARGET
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 SP(NELM,MARR),E(MARR)
      INTEGER*4 J,K,I
      COMMON/MASS/AMASS(109)

C     Construct list of energies

        DE=(EUPPER/ELOWER)**(1./(M-1.))
        E(1)=ELOWER
        DO J=2,M-1
          E(J)=E(J-1)*DE
        END DO
        E(M)=EUPPER

c        OPEN(UNIT=28,STATUS='NEW',FILE='USER:STABLE.DAT')
        stat = creme96_open('stable.dat','user',28,'new')
        WRITE(28,100) ELOWER,EUPPER,M,IZLO,IZUP,TARGET
        WRITE(28,100)
        DO J=IZLO,IZUP
          AA=AMASS(J)
          DO K=1,M
            EE=E(K)
            SP(J,K)=STPOW(EE,FLOAT(J),AA,TARGET)*AA
          END DO
          WRITE(28,200) (SP(J,K),K=1,M)
C         Skip line between elements AJT 5/7/96
          WRITE(28,200)
        END DO
        CLOSE(UNIT=28)

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,2X,1PE10.4)
200   FORMAT((1X,6(1PE10.4,2X)))

      RETURN
      END
```

```
      FUNCTION STPOW(E1,Z0,A1,NAME)
C     ****************************************************************
C     *  THIS ROUTINE RETURNS THE STOPPING POWER OF NUCLIDE (Z0,A1)
C     *  IN MATERIAL 'NAME' AT E1 (MeV/nucleon).
C     *  DATA ON THE STOPPING MATERIAL IS CONTAINED IN TARGET.DAT.
C     ****************************************************************
      CHARACTER*12 NAME$(150),NAME,LNAME
      REAL IADJ$(150,28),NA$(150,28),NASPM$(150,28)
      INTEGER*4 STAT,CREME96_OPEN
      DIMENSION NAS$(150),NZ$(150,28),DENS$(150)
      DIMENSION IGAS$(150),ETAD$(150)
      DATA ITARG,LNAME/0,'QXZ8F3'/
      COMMON /AVG/ AVGZ,AVGZ2,AVGA,AVGI   ! MEAN STOPPING MED. PARAMETERS
C     *  READ IN TARGET DATA
      IF (ITARG.EQ.1) GO TO 100
c     OPEN(UNIT=10,FILE='CREME96:TARGET.DAT',STATUS='OLD',READONLY,SHARED)
      stat = creme96_open('target.dat','cr96tables',10,'old')
 1    FORMAT(1X,I3)
 2    FORMAT(1X,A12,2X,F9.6,2X,F9.6,2X,I1,2X,I2)
 3    FORMAT(1X,I3,2X,F8.4,2X,F5.1,2X,F9.5)
      READ(10,1) NM$
      DO J1=1,NM$
        READ(10,2) NAME$(J1),DENS$(J1),ETAD$(J1),IGAS$(J1),NAS$(J1)
        DO J2=1,NAS$(J1)
          READ(10,3) NZ$(J1,J2),NA$(J1,J2),IADJ$(J1,J2),NASPM$(J1,J2)
        END DO
      END DO
      ITARG=1
      CLOSE(UNIT=10)
 100  CONTINUE

C     *  DETERMINE WHICH TARGET DATA TO USE
      IF (NAME.EQ.LNAME) GO TO 200
      DO J1=1,NM$
        IF(NAME.EQ.NAME$(J1)) K1=J1
      END DO
      LNAME=NAME$(K1)
      IF (NAME.NE.LNAME) THEN
        STPOW=0.
        RETURN
      ENDIF

C     *  COMPUTE MATERIAL PARAMETERS
      RHO=DENS$(K1)
      IGAS=IGAS$(K1)
      ETAD=ETAD$(K1)
      NTOTAL=0
      AVGZ=0.
      AVGZ2=0.
      AVGA=0.
      AVGI=0.
      DO J1=1,NAS$(K1)
        NTOTAL=NTOTAL + NASPM$(K1,J1)
        AVGZ=AVGZ + NASPM$(K1,J1)*FLOAT(NZ$(K1,J1))
        AVGZ2=AVGZ2 + NASPM$(K1,J1)*FLOAT(NZ$(K1,J1))**2
        AVGA=AVGA + NASPM$(K1,J1)*NA$(K1,J1)
        AVGI=AVGI + NASPM$(K1,J1)*ALOG(IADJ$(K1,J1))
      END DO
      AVGZ=AVGZ/FLOAT(NTOTAL)
      AVGZ2=AVGZ2/FLOAT(NTOTAL)
```

```
      AVGA=AVGA/FLOAT(NTOTAL)
      AVGI=EXP(AVGI/FLOAT(NTOTAL))
      RAT=AVGZ/AVGA
 200  CONTINUE

C     *  COMPUTE STOPPING POWERS

      STPOW=0.
      IF (E1.LE.0.) RETURN
      EPRIME=0.6443*Z0+13.7144
      XLAMBDA=(E1/EPRIME)**3.2/1.4427
      IF (XLAMBDA.GT.69.) THEN
        WEIGHT=0.
      ELSE
        WEIGHT=EXP(-XLAMBDA)
      ENDIF
      DO J1=1,NAS$(K1)
        IF (E1.GT.1.) THEN
          TH=RDEDX(E1,Z0,A1,FLOAT(NZ$(K1,J1)),
     &    NA$(K1,J1),IADJ$(K1,J1),1,1,1,1,1,RHO,IGAS,ETAD,RAT)
        ELSE
          TH=0.
          WEIGHT=1.
        ENDIF
        IF (E1.LT.1000.) THEN
          TL=SPLOW(E1,Z0,A1,FLOAT(NZ$(K1,J1)),NA$(K1,J1))
          TN=SPNUC(E1,Z0,A1,FLOAT(NZ$(K1,J1)),NA$(K1,J1))
        ELSE
          TL=0.
          TN=0.
          WEIGHT=0.
        ENDIF
        TTOTAL=TN + WEIGHT*TL + (1.-WEIGHT)*TH
        STPOW=STPOW+NASPM$(K1,J1)*NA$(K1,J1)*TTOTAL
      END DO
      STPOW=STPOW/(AVGA*FLOAT(NTOTAL))
      END

     FUNCTION RDEDX(E1,Z0,A1,Z2,A2,IADJ,I0,I1,I2,I3,I4,
     & RHO,IGAS,ETAD,RAT)
C*****************************************************************
C       AHLEN PROGRAMMED BY SALAMON, MODIFIED BY ADAMS TO
C       INCLUDE RELATIVISTIC BLOCH EFFECT.
C       CHANGES ARE OUTLINED WITH ASTERISKS.
C*****************************************************************
C
C  THIS ROUTINE CALCULATES DE/DX USING THE BETHE EQUATION
C  WITH 3 CORRECTION TERMS, THE MOTT, BLOCH, AND LOW VELOCITY Z**3
C  TERMS. (SEE S.P. AHLEN, PRA17,1236(1978)). ANY OR ALL OF THESE
C  CORRECTIONS CAN BE INCLUDED OR IGNORED, SPECIFIED BY THE
C  INDICES I1(MOTT), I2(BLOCH), I3(LOW VELOCITY Z**3).  A ZERO INPUT
C  FOR A GIVEN PARAMETER ELIMINATES THAT PARTICULAR TERM IN THE
C  DE/DX CALCULATION.
C
C******************************************************************
C  RELATIVISTIC BLOCH CORRECTION (SEE S.P. AHLEN, PRA25,1856(1982))
C  IS CONTROLLED BY THE PARAMETER I4(RELATIVISTIC BLOCH).
C  DENSITY EFFECT IS CONTROLLED BY I0(DENSITY EFFECT).
C******************************************************************
C
```

```
C  THE SHELL CORRECTIONS ARE TAKEN FROM BARKAS AND BERGER,
C  PUBLICATION 1133 OF THE NATL ACAD SCI.  THE LOW VELOCITY Z**3 CORRECTION
C  IS DERIVED FROM A FIGURE IN MCCARTHY AND JACKSON, PHYS REV B6, 4131 (1972).
C  THE MOTT, BLOCH CORRECTIONS ARE FROM AHLEN'S PREVIOUSLY
C  REFERENCED PAPER.
C
C  UNITS OF E1=MEV/AMU
C*********************************************************************
C  Z0 = ATOMIC NUMBER OF STOPPING NUCLEUS
C  Z1 = EFFECTIVE CHARGE OF STOPPING NUCLEUS
C  A1 = ATOMIC MASS OF STOPPING NUCLEUS
C  Z2 = ATOMIC NUMBER OF ELEMENT IN THE STOPPING MEDIUM.
C  A2 = ATOMIC MASS OF ELEMENT IN THE STOPPING MEDIUM
C  RAT=RATIO OF ELECTRONS/MOLECULE TO NUCLEONS/MOLECULE FOR DENSITY
C      EFFECT.
C*********************************************************************
C  UNITS OF IADJ=EV.  IADJ IS A REAL VARIABLE.
C  UNITS OF RETURNED DE/DX=(MEV/AMU)/(G/CM2)
C
C  F1=STANDARD DE/DX FRONT FACTOR
C  F2=STANDARD BETHE NONRELATIVISTIC TERM WITH SHELL CORRECTIONS
C  F3=BLOCH CORRECTION TERM
C  F4=LOW VELOCITY Z**3 NONRELATIVISTIC CORRECTION FACTOR
C  F5=MOTT CORRECTION TERM (Z**3 TO Z**7)
C  F6=STANDARD BETHE RELATIVISTIC TERM
C*********************************************************************
C  F7=RELATIVISTIC BLOCH CORRECTION TERM
C*********************************************************************
C .
C  RHO=DENSITY OF MATERIAL, G/CM**3 (FOR A GAS, GIVE STANDARD DENSITY)
C  IGAS=0 IF CONDENSED PHASE, 1 IF GAS.
C  ETAD=FOR GAS, DENSITY RELATIVE TO STANDARD (1 ATM, 0 DEG CENT). MUST BE >0.
C
C*********************************************************************

      COMMON/FLOOK/F1,F2,F3,F4,F5,F6,F7,Z1

C*********************************************************************
      REAL IADJ
      DIMENSION VA(4),V2FVA(4),Z1ABA(14),COSXA(14)
      DATA Z1ABA,COSXA/0.0,0.05,0.1,0.15,0.20,0.30,0.4,0.5,0.6,
     C  0.8,1.0,1.2,1.5,2.0,1.000,0.9905,0.9631,0.9208,0.8680,
     C  0.7478,0.6303,0.5290,0.4471,0.3323,0.2610,
     C  0.2145,0.1696,0.1261/
      DATA VA,V2FVA/1.,2.,3.,4.,0.33,0.30,0.26,0.23/
      PI=3.14159265
      ALPHA=1./137.03604
      G=1.+E1/931.5016
C*********************************************************************
C  IF I0=0, ELIMINATE DENSITY EFFECT.
        DELT=0.
        IF(I0.EQ.0) GO TO 19
      DELT=DELTA(G,Z2,A2,IADJ,RHO,IGAS,ETAD,RAT)
 19     CONTINUE
C*********************************************************************
      BSQ=1.-1./G**2
      B=SQRT(BSQ)
C*********************************************************************
      Z1=Z0
      TEST=B/Z0**(2./3.)
```

```
      IF(TEST .GT. .1)GO TO 1000
      Z1=Z0*(1.-EXP(-130.*B/Z0**(2./3.)))
 1000 ETA=B*G
C*****************************************************************
      EMASS=0.5110034E+06
      F1=0.3070722*Z1**2*Z2/(BSQ*A2)
      ETAM2=1./ETA**2
      CADJ=1.0E-06*IADJ**2*ETAM2*(0.422377+ETAM2*(0.0304043-ETAM2*
     1   0.00038106))+1.0E-09*IADJ**3*ETAM2*(3.858019+ETAM2*(-0.1667989
     2   +ETAM2*0.00157955))
      F2=ALOG(2.*EMASS*BSQ/IADJ)-CADJ/Z2
      F6=2.*ALOG(G)-BSQ
      F3=0.0
      F4=1.0
      F5=0.0
C*****************************************************************
        F7=0.0
C*****************************************************************
C     IF I2=0, DO NOT CALCULATE BLOCH CORRECTION.
      IF(I2.EQ.0)GO TO 60
      Y=Z1*ALPHA/B
      Y2=Y**2
      MSUM=INT(5.*Y)+1
      SUMR=0.
      DO 90 N=1,MSUM
      FN=FLOAT(N)
      FN2=FN**2
90    SUMR=SUMR+(1./(FN2+Y2)-1./FN2)/FN
      F3=-Y2*(1.202+SUMR)
C
C  IF I3=0, DO NOT CALCULATE LOW VELOCITY CORRECTION.
60    IF(I3.EQ.0)GO TO 50
      V=ETA/(ALPHA*SQRT(Z2))
      IF(V.GE.4.)GO TO 25
      DO 10 I=1,3
      IF(V.GE.VA(I+1))GO TO 10
      V2FV=V2FVA(I)+(V-VA(I))*(V2FVA(I+1)-V2FVA(I))
      GO TO 30
10    CONTINUE
25    V2FV=0.45/SQRT(V)
30    F4=1.+2.*Z1*V2FV/(V**2*SQRT(Z2))
C
C  IF I1=0, DO NOT CALCULATE MOTT CORRECTION.
50    IF(I1.EQ.0)GO TO 70
      Z1A=Z1*ALPHA
      Z1AB=ABS(Z1A/B)
      COSX=0.
      DO 40 I=1,13
      IF(Z1AB.GE.Z1ABA(I+1))GO TO 40
      COSX=COSXA(I)+(Z1AB-Z1ABA(I))*(COSXA(I+1)-COSXA(I))/
     C (Z1ABA(I+1)-Z1ABA(I))
40    CONTINUE
      F5=0.5*Z1A*(B*(1.725+0.52*PI*COSX)+Z1A*(3.246-0.451*BSQ
     1   +Z1A*(1.522*B+0.987/B+Z1A*(4.569-0.494*BSQ-2.696/BSQ
     2    +Z1A*(1.254*B+0.222/B-1.170/BSQ/B)))))
      IF(Z1AB.LE.100.*ALPHA)GO TO 70
C*****************************************************************
      ERR=Z1AB**9/6.
      IF(ERR .GT. .01)ERR=.01
      IF(ERR.LT.ABS(F5/(F2*F4+F3+F5+F6-DELT/2.)))GO TO 70
```

```
C*******************************************************************
      F5=0.
C*******************************************************************
C  IF I4=0, DO NOT CALCULATE RELATIVISTIC BLOCH CORRECTION.
70       IF(I4.EQ.0)GOTO 80
         F7=CR(Z1,G,B)
80    RDEDX=F1*(F2*F4+F3+F5+F6+F7-DELT/2.)/A1
C*******************************************************************
      RETURN

      END


      FUNCTION DELTA(G,Z2,A2,FIADJ,RHO,IGAS,ETA,RAT)
C  THIS FUNCTION IS USED BY RDEDX.
C
C  THIS CORRECTION FOR THE DENSITY EFFECT IS BASED ON STERNHEIMER AND
C  PEIERLS, PHYS REV B3, 3681 (1971).
C  SET IGAS = 0 OR 1,  ETA > 0 REQUIRED.
C  RHO IS DENSITY IN G/CM**3.  FOR A GAS, GIVE RHO AT T=0 DEGREES
C  CENTIGRADE, 1 ATM PRESSURE, AND THE FACTOR ETA WHICH GIVES THE
C  ACTUAL GAS DENSITY UPON MULTIPLICATION BY RHO.
      IF(G.GE.1.8)GO TO 10
      DELTA=0.
      RETURN

C        ************************************************************
10    PLASMA=28.8*SQRT(RHO*RAT) ! RAT REPLACES Z2/A2 FOR MOLECULE
C        ************************************************************
      CBAR=2.*ALOG(FIADJ/PLASMA)+1.0
      B=SQRT(1.-1./G**2)
      Y=2.*ALOG(B*G)+IGAS*ALOG(ETA)
      IF(IGAS.EQ.1)GO TO 100
      IF(FIADJ.GE.100.)GO TO 20
      Y1=9.212
      IF(CBAR.GE.3.681)GO TO 11
      Y0=0.9212
      GO TO 200
11    Y0=1.502*CBAR-4.606
      GO TO 200
20    Y1=13.82
      IF(CBAR.GE.5.215)GO TO 21
      Y0=0.9212
      GO TO 200
21    Y0=1.502*CBAR-6.909
      GO TO 200
100   IF(CBAR.GE.12.25)GO TO 110
      Y1=18.42
      IF(CBAR.LT.12.25)Y0=9.212
      IF(CBAR.LT.11.5)Y0=8.751
      IF(CBAR.LT.11.0)Y0=8.291
      IF(CBAR.LT.10.5)Y0=7.830
      IF(CBAR.LT.10.0)Y0=7.370
      GO TO 200
110   Y1=23.03
      IF(CBAR.GE.13.804)GO TO 120
      Y0=9.212
      GO TO 200
120   Y0=1.502*CBAR-11.52
200   A=(CBAR-Y0)/(Y1-Y0)**3
```

```
      IF(Y.GT.Y0)GO TO 210
      DELTA=0.
      RETURN
 210  IF(Y.GE.Y1)GO TO 220
      DELTA=Y-CBAR+A*(Y1-Y)**3
      RETURN
 220  DELTA=Y-CBAR
      RETURN
      END
```

```
      FUNCTION SIGMA(NU)
C     ***************************************************************
C     **  THIS FUNCTION COMPUTES THE PHASE OF THE GAMMA FUNCTION OF
C     **  1 + i*NU.  IT IS CALLED BY THE FUNCTION CR WHICH COMPUTES
C     **  THE RELATIVISTIC BLOCH CORRECTION.
C     ***************************************************************
      REAL*4 NU
      DIMENSION A(7),H(7)
      DATA(A(I),I=1,5)/.26356,1.4134,3.59642,7.0858,12.6408/
      DATA(H(I),I=1,5)/.521756,.398667,.075942,.003612,23E-6/
      SUM=0.
      DO 1 I=1,5
      SUM=SUM+H(I)*SIN(NU*ALOG(A(I)))
 1    CONTINUE
      DEM=0.
      DO 2 I=1,5
      DEM=DEM+H(I)*COS(NU*ALOG(A(I)))
 2    CONTINUE
      SIGMA=ATAN2(SUM,DEM)
      RETURN
      END

      FUNCTION CR(Z,GAMMA,BETA)
C     ***************************************************************
C     **  THIS FUNCTION COMPUTES THE RELATIVISTIC BLOCH
C     **  CORRECTION TO THE STOPPING POWER ACCORDING TO THE WORK
C     **  OF S.P. AHLEN, PRA25,1856 (1982).
C     ***************************************************************
      REAL*4 LAMBDA,NU,IL1,LNROW,LNROWT
      DATA EULER/.577215665/,PI/3.14159265/
      DATA AMU/931.5016/,ALPHA/.0072973504/
      DATA LAMBDA/1.0/,THETA/0.1/
C   DEFINITIONS
      NU=Z*ALPHA/BETA
      TNU=2.*NU
      DEM=1./(1.+TNU*TNU)
      TNLT=TNU*ALOG(0.5*THETA)
      TNT=TNU*THETA
      TODEM=THETA*DEM
C   COMPUTE REAL AND IMAGINARY PARTS OF L1
      RL1=TODEM*(SIN(TNLT)-TNU*COS(TNLT))
      IL1=-TODEM*(COS(TNLT)+TNU*SIN(TNLT))
      AOBGL=ALPHA/(BETA*GAMMA*LAMBDA)
      TNLOG=TNU*ALOG(AOBGL)
```

```
      COEFF=AOBGL*DEM
      ARG=TNLOG+2.*SIGMA(NU)
C     Bug here discovered by Bonnie Colborn 7-7-95
C     RL1=RL1+COEF*(TNU*COS(ARG)-SIN(ARG))
C     IL1=IL1+COEF*(TNU*SIN(ARG)+COS(ARG))
      RL1=RL1+COEFF*(TNU*COS(ARG)-SIN(ARG))
      IL1=IL1+COEFF*(TNU*SIN(ARG)+COS(ARG))
C   COMPUTE REAL PART OF L2
      FNLOG=2.*TNU*DEM-TNLT
      FNOLOG=(TNU*TNU-1)*DEM+ALOG(0.5*THETA)
      RL2=TODEM*(FNLOG*COS(TNLT)+FNOLOG*SIN(TNLT))
      LNROW=ALOG(2./AOBGL)+EULER-1+(1-TNU*TNU)*DEM
      LNROWT=ALOG(2./AOBGL)+EULER-1+2*DEM
      RL2=RL2+COEFF*(LNROW*SIN(ARG)-TNU*LNROWT*COS(ARG))
C   COMPUTE CR
      PINU=PI*NU
      CR=0.5*PINU*BETA*BETA*(PINU*EXP(PINU)/SINH(PINU))
      CR=CR*(2.*TNU*ALOG(2.)*RL1+(PINU-1.)*IL1+TNU*RL2)
      RETURN
      END


      FUNCTION HYDRGN(EN,Z2,A2)
C     *************************************************************
C     *  STOPPING POWER OF SLOW PROTONS (1 KeV to 1 MeV)
C     *************************************************************
      INTEGER*4 STAT,CREME96_OPEN
      DIMENSION A(92,12)
      DATA MARKER/0/
      IZ2=INT(Z2+0.2)
      IF (Z2.GT.92.) IZ2=92
C     *************************************************************
C     *  ON FIRST CALL TO FUNCTION (MARKER=0) READ IN DATA FROM
C     *  PROTON.DAT.
C     *************************************************************
      IF (MARKER.EQ.0) THEN
c       OPEN(UNIT=50,READONLY,STATUS='OLD',FILE='CREME96:PROTON.DAT',SHARED)
        stat = creme96_open('proton.dat','cr96tables',50,'old')
        DO I=1,92
          READ(50,20) (A(I,J),J=1,11)
20        FORMAT(11(1X,E10.4))
        END DO
        CLOSE(UNIT=50)
        MARKER=1
      ENDIF
      E=EN*1.007825*1000.  !  CHANGE FROM MEV/NUCLEON TO KEV
C     *************************************************************
C     *  COMPUTE STOPPING POWER IN (MEV/NUCLEON)/(G/CM**2)
C     *************************************************************
      IF (E.LE.0.) THEN
        HYDRGN=0.
        RETURN
      ELSE IF (E.LT.1000.) THEN
        SL=A(IZ2,1)*E**.45
        SH=(A(IZ2,2)/E)*ALOG(1.+A(IZ2,3)/E+A(IZ2,4)*E)
        S=SL*SH/(SL+SH)
      ELSE
        G=1.+EN/931.5016
        BSQR=(1.-1./(G*G))
        COEFF=A(IZ2,5)/(BSQR)
```

```
        SHELL=ALOG(A(IZ2,6)*BSQR/(1.-BSQR))-BSQR
        ALOGE=ALOG(E)
        SHELL=SHELL-(A(IZ2,7)+A(IZ2,8)*(ALOGE))
        SHELL=SHELL-(A(IZ2,9)*((ALOGE)**2.)+A(IZ2,10)*((ALOGE)**3.))
        SHELL=SHELL-(A(IZ2,11)*((ALOGE)**4.))
        S=COEFF*SHELL
      ENDIF
      HYDRGN=S*1.E-21/(A2*1.007825*1.659828E-24)
      IF (Z2.GT.92.) HYDRGN=HYDRGN*(Z2**2/92.**2)
      RETURN
      END


      FUNCTION HELIUM(EN,Z2,A2)
C     ***************************************************************
C     *  STOPPING POWER OF SLOW ALPHAS (1 KEV TO 1 MEV)
C     ***************************************************************
      INTEGER*4 STAT,CREME96_OPEN
      DIMENSION A(92,9)
      DATA MARKER/0/
      IZ2=INT(Z2+0.2)
      IF (Z2.GT.92.) IZ2=92
C     ***************************************************************
C     *  ON FIRST CALL TO FUNCTION (MARKER=0) READ IN DATA FROM
C     *  HELIUM.DAT.
C     ***************************************************************
      IF (MARKER.EQ.0) THEN
c       OPEN(UNIT=40,READONLY,STATUS='OLD',FILE='CREME96:HELIUM.DAT',SHARED)
        stat = creme96_open('helium.dat','cr96tables',40,'old')
        DO I=1,92
C          READ(40,20) (A(I,J),J=1,9)
C20        FORMAT(9(1X,E10.4))
           READ(40,20) (A(I,J),J=1,9)
 20        FORMAT(1X,9(E9.4))
        END DO
        CLOSE(UNIT=40)
        MARKER=1
      ENDIF
      E=EN*4.0026*1000.  !  CHANGE FROM MEV/NUCLEON TO KEV

C     ***************************************************************
C     *  COMPUTE STOPPING POWER IN (MEV/NUCLEON)/(G/CM**2)
C     ***************************************************************

      IF (E.LE.0.) THEN
        HELIUM=0.
        RETURN
C  CHANGE IN UPPER ENERGY LEVEL, R.A. WITT 17 MARCH 1994
C     ELSE IF (E.LT.10000.) THEN
      ELSE IF (E.LT.100000.) THEN
        SL=A(IZ2,1)*E**A(IZ2,2)
        SH=(A(IZ2,3)/(E/1000.))*ALOG(1.+A(IZ2,4)/(E/1000.)+A(IZ2,5)
     &  *(E/1000.))
        S=(SL*SH/(SL+SH))
      ELSE
        EE=ALOG(1/(E/1000.))
        S=EXP(A(IZ2,6)+A(IZ2,7)*EE+A(IZ2,8)*EE*EE+A(IZ2,9)*EE*EE*EE)
      ENDIF
```

```
      HELIUM=S*1.E-21/(A2*4.0026*1.659828E-24)
      IF (Z2.GT.92.) HELIUM=HELIUM*(Z2**2/92.**2)
      RETURN

      END



      FUNCTION SPLOW(E,Z0,A1,Z2,A2)

C     ****************************************************************
C     *  STOPPING POWER OF SLOW NUCLEI
C     ****************************************************************

      IZ1=INT(Z0+0.2)
      IF (IZ1.EQ.1) THEN
        SPLOW=HYDRGN(E,Z2,A2)*1.007825/A1
      ELSE IF (IZ1.EQ.2) THEN
        SPLOW=HELIUM(E,Z2,A2)*4.0026/A1
      ELSE
        C=2.99792458E10
        V=C*(1.-(1./(E*1.6022E-6/C/1.673E-24/C+1.))**2.)**0.5
        V1=0.886*(V/2.188E8)*Z0**(-2./3.)
        V2=V1+0.0378*SIN(V1*3.14159265/2.)
        SCALE=(1.-(1.034-0.1777*EXP(-0.08114*Z0))*EXP(-V2))**2.
        SPLOW=SCALE*Z0**2*HYDRGN(E,Z2,A2)*1.007825/A1
      ENDIF
      RETURN

      END


      FUNCTION SPNUC(E,Z0,A1,Z2,A2)

C     ****************************************************************
C     *  LNS STOPPING POWER AS PROGRAMMED BY JR LETAW
C     ****************************************************************

      DATA ZE,A0,EC/4.803242E-10,5.2917706E-9,1.6021892E-6/
      DATA AV,PI,XL/6.022045E23,3.1415927,1.309/
      DATA T1/4.005473E-11/ ! MIN. E = 25 eV
      SPNUC=0.
      IF ((E*A1).LT.2.5E-5) RETURN
      Z=(Z0**(2./3.)+Z2**(2./3.))**1.5
      A=0.8853*A0/Z**(1./3.)
      EL=Z0*Z2*(ZE**2/A)*(A1+A2)/A2
      G=4.*A1*A2/(A1+A2)**2
      ECONV=E*A1*EC
      P1=(2.*XL)**(1./3.)*(ECONV*T1/EL**2/G)**(2./9.)
      P2=(2.*XL)**(1./3.)*(ECONV/EL)**(4./9.)
      FACT=1.125*PI*A**2*(EL**2*G/ECONV)*(AV/A2)/A1/EC
      P1S=SQRT(1.+P1**2)
      P2S=SQRT(1.+P2**2)
      SPNUC=ALOG(P2+P2S)-P2/P2S
      SPNUC=FACT*(SPNUC-(ALOG(P1+P1S)-P1/P1S))
      RETURN
      END
```

```
      SUBROUTINE THIN_SHIELD(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,
     &      PATH,FLUX)

C*********************************************************************
C Special version of UPROPI, for doing transport through thin shield
C without utilizing external files of dE/dx and range-energy, as generally
C done in the UPROP routines.
C
C Important variables
C
C E       Energy at each grid point after shielding
C S       Stopping power at each grid point after shielding
C R       Range at each grid point after shielding
C EP      Energy at each grid point prior to shielding
C SP      Stopping power at each grid point prior to shielding
C RP      Range at each energy EP
C
C*********************************************************************

      IMPLICIT NONE
      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 FLUX(NELM,MARR),E(MARR),FLUX2(MARR)
      REAL*4 R(MARR),S(MARR),EP(MARR),RP(MARR),SP(MARR)
      CHARACTER*12 TARGET
      INTEGER*4 M,IZLO,IZUP,J,K,L,KK,LMAX
      REAL*4 ELOWER,EUPPER,PATH,REL,FUL,DE,XK,AMASS,Z,A
      REAL*4 STPOW
      COMMON/MASS/AMASS(109)
      DATA LMAX/2/

C     Compute vector of energies

      REL=1./ELOWER
      FUL=1./LOG(EUPPER/ELOWER)
      DE=(EUPPER/ELOWER)**(1./FLOAT(M-1))
      E(1)=ELOWER
      DO J=2,M-1
        E(J)=E(J-1)*DE
      END DO
      E(M)=EUPPER

C     Compute range-energy relations and stopping powers:

      DO J=IZLO,IZUP

        Z=FLOAT(J)
        A=AMASS(J)
        CALL RANGE(E,M,Z,A,TARGET,R)
        DO K=1,M
          S(K)=STPOW(E(K),Z,A,TARGET)
        END DO
        DO K=1,M
          DO KK=K,M
            IF (R(KK).GE.R(K)+PATH) GOTO 300
          END DO
          KK=M
300       EP(K)=E(KK)-(R(KK)-R(K)-PATH)*S(KK)
          R(K)=R(K)+PATH
        END DO
```

```
C       Iterate LMAX times to improve estimate of EP

        DO L=1,LMAX
          CALL RANGE(EP,M,Z,A,TARGET,RP)
          DO K=1,M
            SP(K)=STPOW(EP(K),Z,A,TARGET)
            EP(K)=EP(K)-(RP(K)-R(K))*SP(K)
          END DO
        END DO
C
C       Now get flux values at these corresponding external energies:
        DO K=1,M
          XK=1.+(M-1.)*LOG(EP(K)*REL)*FUL
          KK=INT(XK)
          IF (XK.GE.M) THEN
            FLUX2(K)=((EP(K)-E(M-1))*FLUX(J,M)+
     &               (E(M)-EP(K))*FLUX(J,M-1))/(E(M)-E(M-1))
          ELSE
            FLUX2(K)=((EP(K)-E(KK))*FLUX(J,KK+1)+
     &               (E(KK+1)-EP(K))*FLUX(J,KK))/(E(KK+1)-E(KK))
          ENDIF
          FLUX2(K)=FLUX2(K)*SP(K)/S(K)
          IF (FLUX2(K).LT.1.E-20) FLUX2(K)=0.
        END DO
C
        DO K=1,M
          FLUX(J,K)=FLUX2(K)
        END DO

      END DO
      RETURN
      END
```

```
      PROGRAM TRANSPORT_DRIVER

      IMPLICIT NONE
      CHARACTER*80 INFILE,OUTFILE,SHIELDFILE
      CHARACTER*12 TARGET
      INTEGER*4 MARR,NELM
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 INPUT_FLUX(NELM,MARR),OUTPUT_FLUX(NELM,MARR)
      REAL*4 ELOWER,EUPPER,UPATH
      INTEGER*4 M,IZLO,IZUP,IPATH
C
C
C     Get parameters of transport calculation:
C
      CALL INIPROP(INFILE,IPATH,UPATH,TARGET,SHIELDFILE,OUTFILE)

C     Unload input particle flux file into array:
C
      CALL UNLOAD_CREME96_FLUX(INFILE,
     *                         ELOWER,EUPPER,M,IZLO,IZUP,
     *                         INPUT_FLUX)
C
C
C     Now do transport calculation:
C
      CALL CREME96_TRANSPORT(INPUT_FLUX,
     *             ELOWER,EUPPER,M,IZLO,IZUP,
     &             IPATH,UPATH,TARGET,SHIELDFILE,
     &             VERSION_NUMBER,PROGRAM_CODE,
     &             OUTPUT_FLUX)
C
C
C     Now write transported flux to output file:
C
      CALL OUTPUT_TRANSPORTED_FLUX(IZLO,IZUP,ELOWER,EUPPER,
     *                          IPATH,UPATH,TARGET,
     *                          SHIELDFILE,INFILE,
     *                          VERSION_NUMBER,PROGRAM_CODE,
     *                          M,OUTPUT_FLUX,OUTFILE)
C
      STOP
      END
```

```fortran
C***********************************************************************
      subroutine trapped_protons(B, L, yearp, jmod, energy, flux, ne)
C***********************************************************************
c       Inputs:
c           B from Blccoords
c           L from Blccoords
c           yearp from Blccoords
c           jmod,  = 1 for solar min model, = 2 for solar max model
c           energy, an array of values in MeV
c           ne, the number of energy values in the energy array
c       Outputs:
c           flux, ne values of integral flux greater than the corresponding
c                        Mev value in the energy array

C**********************************************************
      implicit none
      save
      real*4 a8max, a8min, B, energy, flux, L, yearp,f1, fil

      integer*4 ie, ifirst, itpfile, jmod, jmodold, l8min, l8max
      integer*4 ne, map
      dimension energy(1), flux(1)     !
      real*8 gmagmo
      common /gmagmo/ gmagmo        !     for esa trara1n
      common /energy/ f1(30,45),fil(30,8)
      common /sumry/ map(777)
      common/ap8min/a8min(8),l8min(16583)
      common/ap8max/a8max(8),l8max(16583)

      integer CREME96_OPEN,stat
      data ifirst /0/
      data itpfile /8/

      if (ifirst .eq. 0 ) then
         ifirst = 1
            jmodold = jmod    !jmod =1 for min =2 for max
            if(jmod .lt. 1 .or. jmod .gt. 2) stop 'tp_model1'
            stat = creme96_open('ap8maxmi.inp','cr96tables',itpfile,
     &                          'old')
c        read in the proton model data to be used
         call modint(itpfile,a8min,l8min)
         read(itpfile,16)
         call modint(itpfile,a8max,l8max)
16       format(19a4)
         close(unit=itpfile)
      endif

  500 continue
      if (jmod .ne. jmodold) then
          type *, ' Model number input has changed '
           stop 'tp_model2'
      endif

      if ((b.eq. 0.) .or. (L.eq. 0.)) return   !not in range of values
      if ( L .gt.11.) return   ! not in range of values

      if (jmod .eq. 1) then
          call trara1(a8min,l8min,L,B,energy,flux,ne)
      elseif (jmod .eq. 2) then
          call trara1(a8max,l8max,L,B,energy,flux,ne)
```

```
      else
          type *, ' Illegal model number input to trapped_protons'
          stop 'tp_model3'
      endif

      do ie = 1, ne
         flux(ie)   =10.**flux(ie)
         if (flux(ie)  .lt.  1.001) flux(ie) = 0.
      enddo
      return
      end
C************************************************************************
      subroutine modint(junit, descr, list)
c************************************************************************
      implicit none
      save
      dimension descr(8), list(16583)
      real *4 descr, dumd

      integer*4 junit, list, length, ic, lnt, i, jc
      integer*4 k, k1, k2, lb, lp,lpp

      equivalence (length,dumd)

      read (junit,1000,end=30) (descr(i),i=1,7),length,lb,ic
      descr(8) = dumd
      type 1002, (descr(i),i = 1,7),length,lb,ic
      lnt = length+1
      lp = lnt/7
      lpp = lp*7
      if (lpp.ne.lnt)lp = lp+1
      lp = lp+1
      k1 = 1
      do jc = 2,lp
         k2 = k1+6
         read(junit,1001,end=30)(list(k),k=k1,k2),lb,ic
         k1 = k2+1
      enddo
      return
   30 type*, ' *** read eof on ',junit,' ***'
      stop 'modint'
 1000 format(2a4,2x,5f10.3,i10,a4,i4)
 1001 format(7i10,a4,i4)
 1002 format(1x,2a4,2x,5f10.3,i10,2x,a4,i4)
      end
C************************************************************************
      subroutine trara1(descr, map, fl, babs, e, f, n)
C************************************************************************
C     B / B0 CASE      JOEL STEIN      9-15-71      X2133      KMS
C   TRARA1 DOES ENERGY VALUE SEARCH FOR FLUX CALCULATION WHEN GIVEN A
C    B AND L POINT.
*  Modified version based on Kluge and Lenhart ESOC Int. Note 78 (1971)
*
*  Modified    APRIL 1988 E.J. DALY ESA/ESTEC/WMA
*                         can pick up geomagnetic dipole moment GMAGMO
*                         from common block and use it in place of
*                         McIlwain's value of .311653
*                         GMAGMO is passed from the geomagnetic
*                         coordinate program.
*                         Default for models should be McIlwain's value
```

```
*
*  Modified    JULY  1993 H.D.R. EVANS    ESA/ESTEC/WMA
*                         Kluge and Lenhart interpolation method in
*                         (B/B0,L) space changed to a linear polygon
*                         interpolation method in (Phi,L) space.
*
C*******************************************************************
C MAP(1) is the first word of list

C S0,S1,S2 are logical variables which indicate whether the flux for a
C particular E,B,L point has already been found in a previous call
C to TRARAP.

      implicit none
      save
      logical s0,s1,s2

      real*4 bob0, babs, descr, e, e0,  e1, e2, f, fl, f0, f1, f2
      real*4 trarap

      integer*4  ie, i0, i1, i2, i3, l3, map, nb, n,  nl

      dimension e(1),f(1),descr(8),map(1)
      real*8 gmagmo
      common / gmagmo / gmagmo

      nl=amin1(32766.,abs(fl*descr(5)))
C  MAX B/B0 ALLOWED HERE IS 1000 (PROTECT AGAINST INTEGER OVERFLOW)

      if (gmagmo .le. 0.) gmagmo = 0.311653

C     BOB0= AMIN1( (BABS*(FL*FL*FL)/ GMAGMO ), 1000.)
      bob0 =  (babs*(fl*fl*fl)/ gmagmo )
      if (bob0 .gt. 1000.) bob0 = 1000.

C     HANDLE CASE WHERE B/B0 IS LESS THAN 1.0 (DISREGARDING REPRESENTATIONAL
c     errors)
      if ( bob0 .lt. 0.95) then
          do  ie = 1, n
             f(ie) = -99.0  ! CHECK TO SEE IF -99 IS USED IN CALLING ROUTINE??????
          enddo
          return
      endif

C     FORCE ANY POSSIBLE REPRESENTATIONAL ERRORS TO 1.0
      BoB0 = AMAX1( BoB0, 1.0)

      nb  = abs((bob0-1) * descr(6))
C  NL IS THE MINIMUM OF THE L VALUE OR 15.999, SCALED TO AN INTEGER BY
C     THE L SCALING FACTOR
C   NB IS THE DIFFERENCE BETWEEN THE INPUT B VALUE AND B EQUATORIAL,
C     SCALED TO AN INTEGER BY THE B SCALING FACTOR.
      i1 = 0
      i2 = map(1)
      i3 = i2+map(i2+1)
      l3 = map(i3+1)
      e1 = map(i1+2)/descr(4)
      e2 = map(i2+2)/descr(4)
      s1 = .true.
      s2 = .true.
```

```
C  I2 IS THE NUMBER OF ELEMENTS IN THE FLUX MAP FOR THE FIRST ENERGY.
C  I3 IS THE INDEX OF THE LAST ELEMENT OF THE SECOND ENERGY MAP.
C  L3 IS THE LENGTH OF THE MAP FOR THE THIRD ENERGY.
C  E1 IS THE ENERGY OF THE FIRST ENERGY MAP (UNSCALED)
C  E2 IS THE ENERGY OF THE SECOND ENERGY MAP (UNSCALED)
C  S1 AND S2 ARE TRUE TO INDICATE THAT NO FLUXES HAVE YET BEEN FOUND.

      do 3 ie = 1,n

C   THE DO STATEMENT LOOPS THROUGH THE ENERGIES FOR WHICH FLUXES ARE
C     DESIRED AT THE GIVEN B,L POINT (BABS,FL).

1     if(e(ie).le.e2.or.l3.eq.0)goto2


C  THE IF STATEMENT CHECKS TO SEE IF THE INPUT ENERGY IS LESS THAN OR E
C   THE ENERGY OF THE SECOND MAP, OR IF THE LENGTH OF THE THIRD MAP IS
C   (I.E. THERE ARE NO HIGHER ENERGIES IN THE TABLE).  IF TRUE, USE TH
C   FOR THOSE TWO ENERGY MAPS TO FIND THE DESIRED FLUX AT THE DESIRED
C   ENERGY.  IF FALSE, THE ZEROTH ENERGY MAP IS DEFINED TO BE TNE FIRS
C   ENERGY MAP, THE FIRST BECOMES THE SECOND, AND THE SECOND BECOMES
C   THE THIRD.  E0,E1,E2 ARE THE ENERGIES FOR THE ZEROTH,FIRST,AND SEC
C   ENERGY MAPS.  F0,F1,F2 ARE THE FLUXES FOR THE ZEROTH, FIRST, AND
C   SECOND ENERGY MAPS AT THE B,L POINT.

      i0 = i1
      i1 = i2
      i2 = i3
      i3 = i3+l3
      l3 = map(i3+1)
      e0 = e1
      e1 = e2
      e2 = map(i2+2)/descr(4)
      s0 = s1
      s1 = s2
      s2 = .true.
      f0 = f1
      f1 = f2
      goto1
 2    if (s1)f1 = trarap(descr,map(i1+1),fl,bob0)
      if (s2)f2 = trarap(descr,map(i2+1),fl,bob0)
C   THESE TWO LOGICAL IFS CALL TRARAP FOR THE FLUX FROM THE FIRST AND
C     SECOND ENERGY MAPS AT THE B,L POINT IF THEY HAVE NOT ALREADY BEEN
      s1 = .false.
      S2 = .FALSE.
C   S1 AND S2 ARE FALSE SINCE F1 AND F2 ARE NOW FOUND.
      f(ie) = f1+(f2-f1)*(e(ie)-e1)/(e2-e1)

C   INTERPOLATE FOR THE FLUX F(IE) USING THE FLUXES AND ENERGIES FOR MAP
C     ONE AND TWO.
C   THE FOLLOWING COMMENTS APPLY TO THE REMAINING PROGRAM STATEMENTS.
C   IF THE FLUX F2 FOR THE SECOND ENERGY MAP IS GREATER THAN ZERO, OR TH
C     ZEROTH ENERGY MAP HAS NOT BEEN DEFINED, THE FINAL FLUX IS THE MAXI
C     OF THE INTEROOLATED FLUX OR ZERO.  IF THE FLUX FOR THE SECOND ENER
C     MAP IS EQUAL TO ZERO, AND THE ZEROTH ENERGY MAP HAS BEEN DEFINED,
C     THEN INTERPOLATE FOR THE FLUX USING THE ZEROTH AND FIRST ENERGY MA
C     CHOOSE THE MINIMUM OF THE TWO INTERPOLATIONS, AND THEN THE MAXIMUM
C     CHOICE AND ZERO FOR THE FINAL FLUX VALUE.
```

```
      if(f2.gt.0.)goto3
      if(i1.eq.0)goto3
      if(s0)f0 = trarap(descr,map(i0+1),f1,bob0)
      s0 = .false.
      f(ie) = amin1(f(ie),f0+(f1-f0)*(e(ie)-e0)/(e1-e0))
 3    f(ie) = amax1(f(ie),0.)
      return
      end

C*************************************************************************
      REAL FUNCTION TRARAP( HEADER, MAP, L, BoB0)
C*************************************************************************
C     PURPOSE:     This function converts the B argument to a hybrid
C                  value, defined by Phi = ASIN( ( B - B0) / (Bmax- B0))
C                  and interpolates the MAP in (PHI-L) space.
C
C     METHOD:      The conversion to the Phi-L space requires the
C                  maximum B value. This is obtained by interpolating
C                  between the maximum B values for  the L strings
C                  that subtend the L value passed to the routine.
C
C                  The Flux at the Phi point along the two L strings
C                  is determined by  locating the three model points
C                  that form a polygon (triangle) that contains the
C                  (L,Phi) point.  The interpolation is then performed
C                  by a linear interpolation using the slope of the
C                  plane in (L,Phi,lnF) space.
C
C     HISTORY:     CREATED July 1993    H.D.R. Evans ESA/ESTEC/WMA
C*************************************************************************
      implicit none
      save
      real  bob0, l, b, phi
      integer*4    map(*)
      real         header(*)
      real         model(3,100,2)
      real         ls(100), bs(100), fluxes(100)
      real         bm(2), bo
      real         verts(3,3)
      real         reqpt(3)
      real         xphi, b0, xinter, bmax, interp
      integer*4      estrng, lstrng
      integer*4      lpos(2), llen, blen(2), li
      integer*4      i, j
      integer*4      s(2), n(2), string, other, tmp
      logical      found, endstr
      logical      inpoly

      if ( bob0 .lt. 1.0 ) then
         trarap = 0.0
         return
      endif

      b = bob0 * 0.311653 / (l**3)       !????ASK TONY: SHOULD WE USE GMAGMO?
      call lvals( map, header, ls, llen)
C     FIND THE TWO L STRINGS THAT SUBTEND THE L VALUE WE ARE GIVEN...
      do li = 1, llen-2
         if ( l .le. ls(li+1) )  go to 6
      enddo
 6    continue
```

```
C     GET THE MAXIMUM B VALUES FOR THE TWO L STRINGS.  ALLOWS US TO
C     LINEARLY INTERPOLATE TO FIND THE MAXIMUM B VALUE FOR THE GIVEN
C     L VALUE.

      do i = 1, 2
         lpos(i) = lstrng( map, header, ls(li+i-1), llen) + 1
         call  bvals ( map( lpos(i)), header,  bs,  fluxes,   blen(i))
         bm(i) = bs(blen(i))
         bo = b0( ls(li+i-1))
         if (bo .gt.0.) then
            do j=1, blen(i)
               model(1,j,i) = ls(li+i-1)
               model(2,j,i) = xphi( bs(j)/bo , bm(i) / bo )
               model(3,j,i) = fluxes(j)
            enddo
         else
            blen(i) = 1
            model(1,1,i) = ls(li+i-1)
            model(2,1,i) = -1.0
            model(3,1,i) = 0.0
         endif
      enddo

      bo = b0( 1)

      phi = xphi( b / bo, xinter( 1, ls(li),   bm(1),
     &                                ls(li+1), bm(2) ) / bo )

C     CHECK FOR AN INVALID PHI VALUE, E.G. IF B > BMAX
      if ( phi .lt. 0 ) then
         trarap = 0.0
         return
      endif

C     IF LENGTH OF BOTH STRINGS IS 1, THEN LINEAR INTERPOLATION BETWEEN
C     POINTS AND RETURN

      if ( blen(1) .eq. 1 .and. blen(2) .eq. 1) then
         trarap =         ( 1 - model(1,1,1)) *
     &                    ( model(3,1,2) - model(3,1,1) ) /
     &                    ( model(1,1,2) - model(1,1,1) )
         if (trarap .gt.0) trarap = ( trarap)
         return
      endif

C     NOW FIND H VALUES IN BOTH L STRINGS THAT SUBTEND REQUIRED POINT.

      reqpt(1) = 1
      reqpt(2) = phi
      s(1) = 1
      s(2) = 1
      n(1) = min( s(1)+1, blen(1))
      n(2) = min( s(2)+1, blen(2))

 10   continue
      endstr = ( s(1) .eq. blen(1)) .and. ( s(2) .eq. blen(2))
      if (.not. endstr) then
         if ( s(1) .eq. blen(1) ) then  !string 1 is empty, have to use string 2 now.
            string = 2
```

```
            other = 1
         else if ( s(2).eq.blen(2) ) then     ! string 2 is empty, have to use string 1.
            string=1
            other =2
         else
            if ( model( 2, n(1), 1) .lt. model(2, n(2),2) ) then
               string = 1
               other = 2
            else
               string = 2
               other = 1
            endif
         endif
         found =  inpoly(  model(1, s(string), string),
     &                        model(1, s(other), other),
     &                        model(1, n(string),string),    reqpt )
         if ( .not. found ) then
            s( string ) = n( string)
            n( string) = min( n(string) +1, blen( string) )
         endif

      endif

      if (.not. (found.or.endstr) ) goto 10

C     REPEAT THIS UNTIL END OF BOTH STRINGS OR POLY FOUND

C     NOW CHECK FOR END OF STRING CONDITION, REQUIRES BACKING UP AND
C     USING A PREVIOUS POINT.

      if ( endstr) then
         if (blen(1) .eq. 1) then
            string = 2
            other = 1
         else if (blen(2) .eq. 1) then
            string = 1
            other = 2
         else if (model(2,s(string)-1,string) .lt.
     &            model(2,s(other)-1,other))
     &   then
            tmp = other
            other = string
            string= tmp
         endif
         n(string)=s(string)-1
      endif

      do j=1,3
         verts(j,1) = model(j, s(string), string)
         verts(j,2) = model(j, s(other), other)
         verts(j,3) = model(j, n(string), string)
      enddo

      trarap = interp( verts, reqpt)
      if (trarap .gt. 0) trarap = ( trarap)

 998  continue
      return
      end
```

```
C*********************************************************************
      real  function interp( gpnts, reqpnt)
C*********************************************************************
C     PURPOSE:    Interpolates between 3 points in 3D space.
C
C     METHOD:     Constructs function of a plane containing the 3 points
C                 and calculates the Z value for the given (X,Y) point.
C
C     HISTORY:    CREATED     July 1993     H.D.R. Evans
C*********************************************************************
      implicit none
      save

      real*4     gpnts( 3, *)
      real*4     reqpnt( 3)
      real*4     v1(3), v2(3)
      real*4     rv(3)
      real*4     plane(4)
      real*4     rv1, rv2, scl
      real*4     dotp, disl2
      integer*4  i

      Interp = 0.0
C     COMPUTE VECTORS IN PLANE OF THREE POINTS.

      do i=1,3
         v1( i) = gpnts(i, 2) - gpnts(i, 1 )
         v2( i) = gpnts(i, 3) - gpnts(i, 1 )
      enddo

C     DETERMINE NORMAL TO PLANE DEFINED BY V1 AND V2, AND PLANE
C     CONSTANT.  PLANE(1)X + PLANE(2)Y + PLANE(3)Z = PLANE(4)

      call CrossP( v1, v2, plane, 3)
      plane(4) = DotP(3, GPnts(1,1), 1, plane, 1)

C     VALUE WE REQUIRE IS THE Z VALUE AT THE POINT SPECIFIED
C     BY THE SOLUTION OF:
C         Z = (PLANE(4) - PLANE(1)X - PLANE(2)Y ) / PLANE(3)

      IF (Plane(3) .NE. 0) THEN
         Interp = (Plane(4) - Plane(1) * ReqPnt(1)
     &                       - Plane(2) * ReqPnt(2) ) / Plane(3)
      else
         print*,'plane containing 3 given points is independent of z'
         print*,'plane = ',plane,char(7)
         stop
      endif

      return
      end

C*********************************************************************
      subroutine crossp( x, y, z, dim)
C*********************************************************************
C     PURPOSE:    Takes the cross product of the two vectors.
C
C     METHOD:     Basic vector calculations.  Vectors must be the
C                 same size.
C
```

```
C     HISTORY:    CREATED     July 1993      H.D.R. Evans ESA/ESTEC/WMA
C***********************************************************************
      implicit none
      save
C     X      - 1ST VECTOR
C     Y      - 2ND VECTOR
C     Z      - CROSSPRODUCT = X^Y
C     DIM    - DIMENSION OF X, Y AND Z

      real*4  x(*), y(*),  z(*)
      real*4  magz
      integer*4     dim
      integer*4     i, j,  indx

      indx (j) = mod( j + dim - 1, dim) + 1

      do i=1, dim
         magz = 0
         z(i) =   x(indx(i+1)) * y(indx(i+2)) -
     &            x(indx(i+2)) * y(indx(i+1))
      enddo
      return
      end

C***********************************************************************
      real*4 function dotp( n, sx, incx, sy, incy )
C***********************************************************************
C     PURPOSE:    Returns the inner (dot) product of SX and SY.
C
C     METHOD:     Basic vector calculations.  Vectors must be the
C                 same size.
C
C     HISTORY:    CREATED     July 1993      H.D.R. Evans ESA/ESTEC/WMA
C***********************************************************************
      implicit none
      save
      integer*4  n, incx, incy
      real*4 sx(*), sy(*)

      integer*4 i, pos

      pos(i,incx) = (i-1)*incx + 1
      dotp = 0
      do i=1,n
         dotp = dotp + sx(pos(i,incx)) * sy(pos(i,incy))
      enddo
      return
      end
C***********************************************************************
      logical function inpoly( a, b, c, pt)
C***********************************************************************
C     PURPOSE:    Returns .TRUE. if the (X,Y) coordinates of point Pt
C                 is in the polygon described by the points A,B,and C
C
C     METHOD:     Pts is decomposed into the sum of the line segments
C                 AC and BC, i.e. PTS = a * AC + b * BC.  If either a
C                 or b is less than zero, then PTS is not subtended by
C                 the lines AC & BC.
C
C                 This is then repeated for the AB and CB line segements
```

```
C                 to ensure PTS isn't the other side of the AB line from C.
C
C     HISTORY:    CREATED     July 1993      H.D.R. Evans ESA/ESTEC/WMA
C
C**********************************************************************
      implicit none
      save
      real*4  a(2), b(2), c(2), pt(2)
      real*4     det
      real*4  m (2,2), im(2,2)
      real*4  u(2), v(2), po(2), coeff(2)
      integer*4  i

      do i=1,2    !  initialise the matrix
         m(1,i) = a(i) - c(i)
         m(2,i) = b(i) - c(i)
         po(i) = pt(i) - c(i)
      enddo

      det = m(1,1) * m(2,2) - m(1,2) * m(2,1)
      if ( det .eq. 0 ) goto 999

      im(1,1) = m(2,2)
      im(2,2) = m(1,1)
      im(1,2) = - m(2,1)
      im(2,1) = - m(1,2)
      coeff(1) = (im(1,1) * po(1) + im(1,2) * po(2))/det
      coeff(2) = (im(2,1) * po(1) + im(2,2) * po(2))/det
      inpoly = ( coeff(1) .ge. 0.0 .and. coeff(2) .ge. 0.0)

C     REPEAT THE PREVIOUS, ONLY THIS TIME WITH THE END POINT.

      do i=1,2
         m(1,i) = b(i) - a(i)
         m(2,i) = c(i) - a(i)
         po(i) = pt(i) - a(i)
       enddo

      det = m(1,1) * m(2,2) - m(1,2) * m(2,1)
      if ( det .eq. 0 ) goto 999

      im(1,1) = m(2,2)
      im(2,2) = m(1,1)
      im(1,2) = - m(2,1)
      im(2,1) = - m(1,2)
      coeff(1) = (im(1,1) * po(1) + im(1,2) * po(2))/det
      coeff(2) = (im(2,1) * po(1) + im(2,2) * po(2))/det
      inpoly = ( coeff(1) .ge. 0.0 .and. coeff(2) .ge. 0.0)
     &          .and. inpoly
      return

 999  continue
      inpoly = .false.
      print*,char(7)
      stop '***inpoly*** determinant = 0'
      end
C**********************************************************************
      real*4  function xinter( x, x1, y1, x2, y2)
c**********************************************************************
c     purpose:    linearly interpolates between (x1,y1) and (x2,y2).
```

```
c
c     method:     simple linear interpolation.
c
c     history:    created     july 1993      h.d.r. evans esa/estec/wma
c**********************************************************************
      implicit none
      save
      real*4  x, x1, x2, y1, y2

      if ( x2 .ne. x1 ) then
         xinter = y1 + (x- x1)*(y2-y1)/(x2-x1)
      else
         xinter = y1
      endif

      return
      end
c**********************************************************************
      integer*4 function estrng( map, header, e, len)
c**********************************************************************
c
C     RETURNS INDEX IN THE AX8 MAP WHERE REQUESTED ENERGY STRING STARTS.

c**********************************************************************
      implicit none
      save
      real*4     e
      real*4     header(*), energy
      integer*4  index, len
      integer*4  map(*)
      integer*4 epos, escl, maplen
      data    epos, escl, maplen / 1, 4, 8/

c     len   : length of current energy string
c     epos  : offset in the energy string of the energy
c     maplen: position in the header of the total map length
c     escl  : position in the header of the energy scale factor

      index = 1
      energy = 0

 10   if ( (e.le.energy) .or. (index .gt. header(maplen) ) ) goto 20
         len = map( index)
         energy   = 1.0 * map( index + epos) / header(escl)
         if ( e .gt. energy) index = index + len
      goto 10
 20   continue
      estrng = index
      return
      end
C**********************************************************************
      integer*4  function lstrng( estr, header, l, len)
C**********************************************************************C
C     Returns the index in the Energy string (ESTR)
C     that the requested L string starts.
C**********************************************************************
      implicit none
      save
      real*4     l
      real*4     header(*)
```

```
      real*4     mapl
      integer*4  index, len
      integer*4  estr(*)
      integer*4 slen, lpos, lscl
      data    slen, lpos, lscl   /   1,    1,    5 /

c     slen  - position in the e string of the e string length
c     lpos  - offset in the l string of the l value
c     lscl  - position in the header of the l scale factor
c     index = position of the first l string in the e string
      index = 3

      mapl = 0

      if ( l .eq. 0 ) then
         lstrng = index - 1
         return
      endif

 10   if ( (l.le.mapl) .or. (index .gt. estr(slen) ) ) goto 20
         len = estr( index)
         mapl = estr( index + lpos) / header( lscl)
         if ( l .gt. mapl)   index = index + len
      goto 10
 20   continue

      lstrng = index - 1
      return
      end

C*************************************************************************
      subroutine  evals(  map, header, e, npts)
C*************************************************************************
C     Searches through the Ax8 model for all of the energies it contains.
C*************************************************************************
      implicit none
      save
      real*4          header(*)
      real*4          e(*)
      integer*4       map(*)
      integer*4       npts, index
      integer*4  epos, escl, maplen
      data     epos, escl, maplen / 1, 4, 8/

C     EPOS     : OFFSET IN THE ENERGY STRING OF THE ENERGY
C     ESCL     : POSITION IN THE HEADER OF THE ENERGY SCALE FACTOR
C     MAPLEN   : POSITION IN THE HEADER OF THE TOTAL MAP LENGTH

      index    = 1
      npts     = 0

 10   continue
         npts = npts + 1
         e(npts) = map( index + 1) / header( escl)
         index = index + map(index)
      if ( index .le. header(maplen) .and. map(index) .ne. 0) go to 10

      return
      end
C*************************************************************************
```

```
      subroutine  lvals(  estr, header, l, npts)
C*********************************************************************
C     Searches through Energy string for all of the L values it contains.
C*********************************************************************
      integer*4     estr(*)
      real*4        header(*)
      real*4        l(*)
      integer*4     npts

C     SLEN  - POSITION IN THE E STRING OF THE E STRING LENGTH
C     LPOS  - OFFSET IN THE L STRING OF THE L VALUE
C     LSCL  - POSITION IN THE HEADER OF THE L SCALE FACTOR

      integer*4 slen, lpos, lscl
      data    slen, lpos, lscl /   1,    1,    5 /

      index    = 3
      npts     = 0
 10   if ( index .ge. estr(slen) ) goto 20
         npts = npts + 1
         l(npts) = estr( index + lpos) / header( lscl)
         index = index + estr( index)
      goto 10
 20   continue
      return
      end
C*********************************************************************
      subroutine  bb0val( lstr, header, bob0, lnflux, npts)
C*********************************************************************
C     Returns the BoBo and Log10(flux) values contained in the L string
C     (MAP)
C*********************************************************************
      implicit none
      save
      integer*4     lstr(*)
      real*4        header(*)
      real*4        bob0(*)
      real*4        lnflux(*), i
      integer*4     npts
      integer*4     len
      integer*4 slen, bscl, flxscl, flxinc, flxoff
      data    slen, bscl, flxscl, flxinc, flxoff
     &        /   1,    6,     7,  -256,    3 /

C     SLEN  - POSITION IN THE B STRING OF THE B STRING LENGTH
C     LPOS  - OFFSET IN THE L STRING OF THE L VALUE
C     LSCL  - POSITION IN THE HEADER OF THE L SCALE FACTOR
C     FLXSCL - POSITION IN THE HEADER OF THE LNFLUX SCALE FACTOR
C     FLXINC  - UNSCALED INCREMENT IN B BETWEEN STRING VALUES
C     FLXOFF  - OFFSET IN B STRING OF THE B0 LNFLUX VALUE

      npts  = 1
      len  = lstr( slen)
      bob0(1)  = 1.0
      lnflux(1) = lstr( flxoff) / header(flxscl)
      if ( len .lt. 4) return

      npts = 0
      i = 4
 10   if ( (i.gt.len) .or. (lstr(i).le. 0) ) goto 20
```

```
         npts = npts + 1
         lnflux(i-2) =  (lstr(flxoff) + flxinc*(i-flxoff)) /
     &                   header(flxscl)
         bob0(i-2)   = bob0(i-flxoff) + lstr(i)/header(bscl)
         i = i + 1
      goto 10
 20   continue
      return
      end


C*********************************************************************
      subroutine  bvals(  lstr, header, b, lnflux, npts)
C*********************************************************************
c    Returns B and Log10(flux) values contained in L string (MAP)
C*********************************************************************
      implicit none
      save
      real*4         header(*),   b(*)
      real*4         lnflux(*),  bob0(40)
      real*4         l, bo, b0
      integer*4      lstr(*)
      integer*4      npts, i
      integer*4  lpos, lscl
      data     lpos, lscl /  2,  5/

C     LPOS  - POSITION IN L STRING OF THE L VALUE
C     LSCL  - POSITION IN THE HEADER OF THE L SCALE FACTOR

      l     = lstr(lpos) / header( lscl)
      call  bb0val( lstr, header, bob0, lnflux, npts)
      bo = b0( l)
      do i = 1, npts
         b(i) = bob0(i) * bo
         if ( b(i) .eq. 0)   lnflux(i) = 0.0
      enddo
      return
      end
C*********************************************************************
      subroutine  phival(  lstr, header, l, phi, lnflux, npts)
C*********************************************************************
C     Returns Hybrid and Log10(flux) values contained in L string (MAP)
C*********************************************************************
      implicit    none
      save
      real*4         bm, b0(100), bmax
      real*4         header(*)
      real*4         l
      real*4         lnflux(*)
      real*4         phi(*)
      real*4         xphi
      integer*4      i
      integer*4      lstr(*)
      integer*4      npts

      call bb0val( lstr, header, b0, lnflux, npts)
C     CONVERT THE B VALUES TO THE HYBRID ONES
      bm = b0( npts)
      do 10 i = 1, npts
         phi(i) = xphi( b0(i), bm)
```

```
         if ( phi(i) .lt. 0) lnflux(i) = 0.0
 10   continue
      return
      end
C***********************************************************************
      real*4  function xphi ( bob0, bmax)
C***********************************************************************
c    Computes the hybrid magnetic field coordinate=
C               ( B/B0 - 1      )
C     XPHI= ASIN( --------      )
C               ( Bmax/B0 - 1  )
C     Where Bmax is the atmospheric cutoff value for B.
C     If Bmax = 1, then XPHI = 90. ( Arcsin( 1.0) )
C***********************************************************************
      implicit none
      save
      real*4  bob0, bmax
      real*4  sine

      if ( bob0 .gt. bmax) then
C        ARCSIN( >1.0) --- BOB0 BEYOND THE ATMOSPHERIC CUT OFF.
         xphi = -1.0
         return
      endif
      if ( bmax .ne. 1) then
         sine = (bob0 - 1)/(bmax-1)
         if ( (-1.le.sine).and.(sine .le.1.0) ) then
            xphi = asin( sine ) * 180.0 / 3.1415927
         else
            xphi = -1.0
         endif
      else
         xphi = -1.0
      endif
      return
      end
C***********************************************************************
      real*4  function b0(l)
C***********************************************************************
C     computes magnetic field strength for an L shell at magnetic equator.
C***********************************************************************
      implicit none
      save
      real*4  l
      if ( l .gt. 0 ) then
         b0 = 0.311653 / (l*l*l)     !   ????ASK TONY: SHOULWE USE GMAGMO?
      else
         b0 = 0
      endif
      return
      end
```

```
      PROGRAM Trapped_Proton_Driver
C
      IMPLICIT NONE

      SAVE  !from original DRIVER supplied by Colborn & Armstrong
C
     integer*4 ie, iemax, ifile, igo, imod,  ne, ns, Mpts

     parameter (iemax = 30)    ! MAX NO. OF ENERGIES ALLOWED
     real Evals(iemax)

      INTEGER NLvals
      PARAMETER (NLvals=10)

      INTEGER Ndays    !program now calculates this from period and Norbits.
      INTEGER Norbits,Norbsteps   !arguments to subroutine calls, 11-26-97.

      REAL OrbPrecTime  !for future use, 11-26-97.

      REAL TrappedFlux(IEmax,NLvals),XLbounds(NLvals)
      REAL RelDwellTime(NLvals)
      REAL Year
      REAL DiffTrappedFlux(IEmax,NLvals)

      INTEGER MARR
      PARAMETER (MARR=5000)
      REAL Eout(MARR),Fluxout(MARR,NLvals)

      REAL OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp

      INTEGER ILbins
      CHARACTER*80 TrappedFile

      INTEGER Program_Code
      DATA Program_Code/1/

      INTEGER IpreCalc
      LOGICAL DistBelt,PreCalcFlux

      REAL ELOWER,EUPPER
      DATA ELOWER,EUPPER/1.0E-01,1.0E+05/

      REAL OrbPeriod
C--------------------------------------------------------------------------------

      CALL TrappedDriverInput(Evals,NE,OrbIncl,Apogee,Perigee,
     #        AscNodeLong,AscNodeDisp,PerigDisp,TrappedFile,Year,
     #        XLbounds,ILbins,imod,DistBelt,PreCalcFlux,IPreCalc,
     #        Ndays,Norbits,Norbsteps)

      CALL Trapped_ORBINT(OrbIncl,Apogee,Perigee,AscNodeLong,
     #        AscNodeDisp,PerigDisp,Evals,NE,TrappedFlux,
     #        DiffTrappedFlux,Year,XLbounds,ILbins,imod,RelDwellTime,
     #        DistBelt,PreCalcFlux,IPreCalc,
     #        Ndays,Norbits,Norbsteps,OrbPeriod,OrbPrecTime)

      CALL Trapped_Spectra(Evals,NE,ELOWER,EUPPER,Mpts,ILbins,
     #        TrappedFlux,DiffTrappedFlux,Eout,Fluxout)

      CALL OutputTrappedFlux(Evals,NE,TrappedFlux,DiffTrappedFlux,
```

```
     #          TrappedFile,OrbIncl,Apogee,Perigee,AscNodeLong,
     #          AscNodeDisp,PerigDisp,Year,XLbounds,ILbins,PROGRAM_CODE,
     #          imod,RelDwellTime,DistBelt,PreCalcFlux,IPreCalc,
     #          ELOWER,EUPPER,Mpts,Eout,Fluxout,
     #          Ndays,Norbits,Norbsteps,OrbPeriod,OrbPrecTime)

        STOP
        END

c***********************************************************************
      subroutine differ(ne, e, fa, fb, d, Ilbins)
c***********************************************************************

      implicit none
      save
      real*4  da, db
      integer*4 i, j, ne, ILbins, L

      INTEGER iemax,NLvals
      PARAMETER (iemax = 30,NLvals=10)

C     make array fixed size, so that don't have difficulties with 2D
C     array allignment.

      REAL*4 e(iemax),fa(iemax,NLvals)
      REAL*4 fb(iemax,NLvals),d(iemax,NLvals)

c     routine finds d(i) the differential flux at energy e(i)
c     assuming that the spectrum is best represented by an exponential
c
c     ne is number of energies
c     fa(i) is integral flux above e(i)
c     fa(ne) is defined by routine
c     fb(i) is the flux between e(i+1) and e(i)

C-----------------------------------------------------------------------

      do L = 1,ILbins

      do i = ne,1,-1
         if (fa(i,L) .gt. 0) go to 2
         d(i,L) = 1 .0e-37
      enddo
      GOTO 99

2     if (i .eq. ne) i = i-1
      fa(i+1,L) = fa(i,L) - fb(i,L)
      if (fa(i+1,L) .ne. 0) go to 6
      i = i-1
      if (i .gt. 1) go to 6
      d(1,L) = fa(1,L)/(e(2)-e(1))
      GOTO 99

6     db = -alog(fa(2,L)/fa(1,L))/(e(2)-e(1))*fa(1,L)
      do j = 1,i
         da = -alog(fa(j+1,L)/fa(j,L))/(e(j+1)-e(j))*fa(j,L)
C        Added error checking on da*db, 11-24-97.
         IF (da*db .GE. 0.0) THEN  !da*db should be >= 0 for physical solutions
           d(j,L) = sqrt(da*db)
         ELSE
```

```
            d(j,L) = 0.0
          ENDIF
          db = da*fa(j+1,L)/fa(j,L)
      enddo

      d(i+1,L) = da*fa(i+1,L)/fa(i,L)    ! for protons
99    CONTINUE  !for going to next L-value bin instead of return
      enddo  !stepping through L-bins.
      return
      end

C*********************************************************************

        SUBROUTINE TrappedDriverInput(Evals,NE,OrbIncl,Apogee,Perigee,
     #                 AscNodeLong,AscNodeDisp,PerigDisp,TrappedFile,
     #                 Year,XLbounds,ILbinsum,imod,
     #                 DistBelt,PreCalcFlux,IPreCalc,
     #                 Ndays,Norbits,Norbsteps,OrbPrecTime)

        IMPLICIT NONE

        REAL OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp

C       Note that the eccentricity is calculated here to decide if
C       need to read PerigDisp.  The eccentricity is also recalculated
C       in the initialization CALL ORBIT(1,...) case.
C
C       This makes the input driver independent of the actual computational
C       routines, so that it will be easier to modify and interface with other
C       space environment routines.

        REAL E,Re     !eccentricity and radius of Earth
        PARAMETER (Re=6371.2)
        REAL ApPerSwitch

        INTEGER IERR,IACCEPT
        DATA IERR/0/

        INTEGER Ndays
        INTEGER Norbits,Norbsteps   !now passed as arguments, 11-26-97.

        REAL OrbPrecTime  !for future use, 11-26-97.

        CHARACTER*80 TrappedFile

        INTEGER NLvals,I,L,ILbinMax,ILbinsum,imod
        PARAMETER (NLvals=10)

        REAL XLbounds(NLvals),XLinfinite,Year,YearMin,YearMax
        PARAMETER (XLinfinite=1.0E+06)

C       Appropriate epochs for AP8MIN & AP8MAX L-value calculations
        PARAMETER(Yearmin=1964.0,Yearmax=1970.0)

        REAL XLdummy

      INTEGER iemax
      parameter (iemax = 30)    ! MAX NO. OF ENERGIES ALLOWED
      real evals(iemax)
      integer ne
```

```
c     define energy grid for test case
c     energy values often used in space station calculation
C     These are hardwired into calculation for now.
C     Changed E to Evals (since E is the now the eccentricity)

        INTEGER IpreCalc,IFLUXtype
        LOGICAL DistBelt,PreCalcFlux
        INTEGER Idummy  !For diagnostic message on number of I/O parameters.
C-----------------------------------------------------------------------

        NE = 29
        EVALS(1)=  1.0
        EVALS(2)=  2.0
        EVALS(3)=  3.0
        EVALS(4)=  4.5
        EVALS(5)=  6.
        EVALS(6)=  8.
        EVALS(7)=  10.
        EVALS(8)=  15.
        EVALS(9)=  20.
        EVALS(10)=  30.
        EVALS(11)=  50.
        EVALS(12)=  80.
        EVALS(13)=  100.
        EVALS(14)=  150.
        EVALS(15)=  200.
        EVALS(16)=  250.
        EVALS(17)=  300.
        EVALS(18)=  350.
        EVALS(19)=  400.
        EVALS(20)=  450.
        EVALS(21)=  500.
        EVALS(22)=  550.
        EVALS(23)=  600.
        EVALS(24)=  800.
        EVALS(25)=  1000.
        EVALS(26)=  3000.
        EVALS(27)=  1.0E+04
        EVALS(28)=  3.0E+04
        EVALS(29)=  1.0E+05

       WRITE(6,1000)
       WRITE(6,1001)

C       initialize boundaries L-value bins

        XLbounds(1)=0.0

        DO L=2,NLvals
          XLbounds(L)=XLinfinite
        ENDDO

 9390   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,390)
        IFLUXtype=0
        READ(*,*,ERR=9390,IOSTAT=IERR)IFLUXtype   !installed, 11-26-97.

        PreCalcFlux=.FALSE. !initialize not to use pre-calculated Flux
```

```
                             !hardwired off for now.
        DistBelt=.FALSE.     !hardwire off for now.
        IpreCalc=0

        IF (IFLUXtype .NE. 0) THEN
 9391     CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(*,391)
          READ(*,*,ERR=9391,IOSTAT=IERR)IpreCalc
          PreCalcFlux=.TRUE.
          DistBelt=.FALSE.     !hardwire off for now, since we do not include a
                               !"disturbed belt" model

C         Use quiet-time, 51.6 degrees as the default case.

          IF (IpreCalc .LT. 0 .OR. IpreCalc .GT. 3) IpreCalc=0

 9427     CONTINUE
          CALL RETRY_INPUT(IERR)
          WRITE(*,427)
          READ(*,428,ERR=9427,IOSTAT=IERR)TrappedFile
          CALL CHECK_OUTPUT_FILE(TrappedFile,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 9427

C         The SUBROUTINE OutputTrappedFlux  presently takes these variables
C         from the header information in the pre-calculated files.
C         IMOD is also taken from that header information.  Only IpreCalc
C         is re-checked against that header information.  12-1-97.
C
CF?            IF (IpreCalc .EQ. 0 .OR. IpreCalc .EQ. 1) THEN
CF?              OrbIncl=51.6
CF?              Apogee=400.0
CF?              Perigee=400.0
CF?            ELSEIF (IpreCalc .EQ. 2 .OR. IpreCalc .EQ. 3) THEN
CF?              OrbIncl=28.5
CF?              Apogee=450.0
CF?              Perigee=450.0
CF?            ENDIF

C         The pre-calculated fluxes are not available (and not divided into
C         L-bins)
          ILbinsum=1

          RETURN

        ENDIF

C       Moved the questions and answers on AP8MIN or AP8MAX here, 12-11-97.

 9454   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,454)
        READ (*,*,ERR=9454,IOSTAT=IERR) imod
        IF (imod .LT. 0 .OR. imod .GT. 1) imod=0

      imod = imod + 1    !   model no. for subroutine blccoords
                         !   model no. for subroutine trapped_protons

        IF (imod .EQ. 1) YEAR=1964.0
        IF (imod .EQ. 2) YEAR=1970.0
```

```
C       End of moved questions & answers on AP8MIN or AP8MAX, 12-11-97.

C       What is the altitude at apogee?
C
 9420   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,420)
        READ (*,*,ERR=9420,IOSTAT=IERR)  Apogee
C
C       WHAT IS THE ALTITUDE AT PERIGEE?
C
 9400   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,400)
        READ (*,*,ERR=9400,IOSTAT=IERR)  Perigee

C       allow the user to specify apogee and perigee in either order
C       instead of performing unintended calculation which sets eccentricity
C       to zero and using Perigee variable (actual apogee) to produce
C       a circular orbital altitude in ORBIT routine.

        IF (Perigee .GT. Apogee) THEN
           ApPerSwitch=Apogee
           Apogee=Perigee
           Perigee=ApPerSwitch
           WRITE(*,430)
        ENDIF

        E=(Apogee-Perigee)/(Apogee+Perigee+2.*Re)
        IF (E.LT..00001) E=0.
C
C       WHAT IS THE ORBITAL INCLINATION?
C
 9405   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,405)
        READ (*,*,ERR=9405,IOSTAT=IERR)  OrbIncl

C       READ in number of orbits, 11-26-97.

C       Have Removed "FAST" option, i.e. must enter Ascending Node information
C
C       Retain these initializations in case want to hardwire ascending
C       node information at future time.

        AscNodeLong=0.
        AscNodeDisp=0.
        PerigDisp=0.

C       Modified WRITE statement, 12-11-97.

        IF (E.NE.0.) THEN
          Idummy=3
        ELSE
          Idummy=2
        ENDIF

        WRITE(*,409)Idummy
```

```
C       WHAT IS THE INITIAL LONGITUDE OF THE ASCENDING NODE?
C

 9410   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,410)
        READ (*,*,ERR=9410,IOSTAT=IERR) AscNodeLong
C
C       WHAT IS THE INITIAL DISPLACEMENT FROM THE ASCENDING NODE?
C
 9415   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,415)
        READ (*,*,ERR=9415,IOSTAT=IERR)  AscNodeDisp

        IF (E.NE.0.) THEN       !Only read in XI if eccentricity is nonzero
C
C           What is the displacement of the perigee from the ascending node?
C
 9425    CONTINUE
         CALL RETRY_INPUT(IERR)
         WRITE(*,425)
         READ (*,*,ERR=9425,IOSTAT=IERR)  PerigDisp

        ENDIF

        IF ( (AscNodeLong .NE. 0.0) .OR. (AscNodeDisp .NE. 0.0) .OR.
     &       (PerigDisp .NE. 0.0) ) WRITE(*,426)

C       Moved the entry of Norbits here, 12-11-97.

 9408   CONTINUE

        CALL RETRY_INPUT(IERR)
        WRITE(*,408)
CF?        READ (*,*,ERR=9408,IOSTAT=IERR) Norbits,Norbsteps
        READ (*,*,ERR=9408,IOSTAT=IERR) Norbits
        Norbsteps=200     !hardwired at 200 for now, 11-26-97.
        IF (Norbits .LE. 0) Norbits=200
CF?        IF (Norbsteps .LE. 0) Norbsteps=200

C       End of moved section for entry of Norbits, 12-11-97.

 9450   CONTINUE
        CALL RETRY_INPUT(IERR)
        WRITE(*,450)
        READ (*,*,ERR=9450,IOSTAT=IERR)  ILbinMax

        IF (ILbinMax .LT. 0) ILbinMax=0

        IF (ILbinMax .GT. NLvals) THEN
           WRITE(*,456)
           ILbinMax = NLvals
        ENDIF

        IF (ILbinMax .GT. 0) THEN
          WRITE(*,451) ILbinMax
 9451       CONTINUE
            CALL RETRY_INPUT(IERR)
            READ (*,*,ERR=9451,IOSTAT=IERR)  (XLbounds(L),L=1,IlbinMax)
```

```
            IF (ILbinMax .EQ. 1 .AND. XLBOUNDS(1) .EQ. 0.0) WRITE(*,458)
        ENDIF

        IF (XLbounds(1) .LT. 0.0) XLbounds(1)=0.0

C       Start DO loop at 1, so that ILbinMax=2 will be properly handled
C       This SUBROUTINE insists the L-values are in increasing order.
C       If this is not the case, all subsequent L-value bins will be
C       ignored.

        DO L=1,ILbinMax
          IF (XLbounds(L) .LT. XLbounds(1)) THEN
              WRITE(*,452) XLbounds(L),XLinfinite
              XLbounds(L)=XLinfinite
          ENDIF
          IF (L .GE. 2) THEN
            IF (XLbounds(L) .LE. XLbounds(L-1)) THEN
              WRITE(*,452) XLbounds(L),XLinfinite
              XLbounds(L)=XLinfinite
            ENDIF
          ENDIF
        ENDDO

        ILbinsum=1

        DO L=1,ILbinMax
          IF ( (L .GE. 2) .AND. (XLbounds(L) .LT. XLinfinite) )
     &          ILbinsum=ILbinsum+1
        ENDDO

        IF (ILbinMax .NE. ILbinsum .AND. ILbinMax .NE. 0) THEN
          WRITE(*,453)ILbinMax,ILbinsum
          ILbinMax=ILbinsum
        ENDIF

 9428   CONTINUE
        CALL RETRY_INPUT(IERR)
        IF (ILbinMax .EQ. 0 .OR. (ILbinMax .EQ. 1 .AND.
     &      XLBOUNDS(1) .EQ. 0.0) ) THEN
          WRITE(*,427)
          READ(*,428,ERR=9428,IOSTAT=IERR)TrappedFile
          CALL CHECK_OUTPUT_FILE(Trappedfile,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 9428
        ELSE
          WRITE(*,455)ILbinMax
          READ(*,428,ERR=9428,IOSTAT=IERR)TrappedFile
          CALL CHECK_OUTPUT_FILE(Trappedfile,IACCEPT)
          IF (IACCEPT.NE.0) GOTO 9428
          DO I=1,LEN(TrappedFile)
            IF (TrappedFile(I:I) .EQ. '.') THEN
                TrappedFile=TrappedFile(1:I-1)
            ENDIF
          ENDDO
        ENDIF

        RETURN


 1000  FORMAT(1X,'Orbit averaging using ESA AP8 Models.',/)
 1001  FORMAT(' This program will calculate the omnidirectional',
```

```
     &        ' trapped proton flux',
     &      /,' to a',
     &        ' spacecraft orbiting inside the magnetosphere.  The',
     &      /,' calculated trapped flux is for demonstration purposes',
     &        ' only, and is NOT part of the standard CREME96 software.',
     &//,' NOTE: Before running this or any other CREME96 programs',
     &  ' please define three ',
     & /,' logicals: ',/,
     & /,4x,' CREME96       as the directory where CREME96 source',
     &        ' & executables reside.',
     & /,4x,' CR96TABLES    as the directory in which CREME96 data',
     &        ' tables reside.',
     & /,4x,' USER          as the directory in which output files',
     &        ' should be written.',
     &  //,' Now begin specification of the ESA AP8 calculation: ',/)

390   FORMAT(1X,'Enter 0 to directly calculate the trapped ',
     &          'proton fluxes, or',/,1X,'Enter 1 to test the ',
     &          'pre-calculated trapped proton interface.',/,3X,
     &         '[NOTE:  The pre-calculated fluxes are presently ',
     &         'test case results which',/,11X,
     &         'are intended only as a software test.]')

391   FORMAT(1X,'Enter 0 for Space Station (51.6 deg., 400 km)',
     &      ' orbit (ISSA), ESA-AP8MAX,',
     &  /,7X,'1 for ISSA, ESA-AP8MIN,',
     &  /,7X,'2 for 28.5 deg. (450 km), ESA-AP8MAX,',
     &  /,7X,'3 for 28.5 deg. (450 km), ESA-AP8MIN.')

420   FORMAT(/,1X,'Enter altitude at apogee (kilometers):  ')
400   FORMAT(/,1X,'Enter altitude at perigee (kilometers):  ')
430   FORMAT(/,1X,'Input apogee < perigee, have been interchanged.')

405   FORMAT(/,1X,'Enter orbital inclination (degrees):  ')

c     Modified 12-11-97.
409   FORMAT(/,1X,'The following ',I1,' input parameters are ',
     &    'most relevant to situations in which',
     &  /,1X,'the actual orbital path is known',
     &       ' or in which mission critical operations',
     &  /,1x,' are planned.',
     & //,3X,'[Recommended values are 0.0, unless you wish to examine',
     &  /,3X,'a very specific orbital segment.]')

410   FORMAT(/,1X,'Enter initial longitude of ascending node',
     &          1X,'[Recommended = 0.0 (degrees)]:')
415   FORMAT(/,1X,'Enter initial displacement from ascending',
     1          ' node',1X,'[Recommended = 0.0 (degrees)]:')
425   FORMAT(/,1X,'Enter displacement of perigee from',
     1  ' ascending node',1X,'[Recommended = 0.0 (degrees)]:')

426   FORMAT(/,1X,'Note:  for studies sensitive to a specific',
     &  ' orbital segment, you should be',
     &  /,1X,'aware that the trapped proton',
     &  ' calculations are averaged over 7 days at present.  This',
     &  /,1X,'parameter can be easily reset by modifying',
     &  ' the TRAPPED_ORBINT subroutine, but is ',
     &  /,1x,'not provided as a general-use input parameter.')
```

```
 427    FORMAT(/,1X,'Enter name of output trapped proton file:',
     &        /,'    [Recommended:  something.TRP]')
 428    FORMAT(A80)

 450    FORMAT(/,1X,'Enter the number of desired TRP L-value bins ',
     &       '(1 - 10):',
     &  /,3X,'[Recommended default = 0, i.e. ',
     &       'one trapped flux calculation for the entire orbit.]')

 451    FORMAT(/,1X,
     &       'Enter the lower limits of the ',I2,' L-value bins: ',
     &  /,3X,'[A typical scenario could be to request 4 bins as the',
     &       ' as the previous entry.',
     &  /,3X,'Then, entries of 0.0, 2.0, 4.0, and 6.0',
     &       ' would subdivide the orbit into',
     &  /,3x,'sections with L < 2, L = 2-4, L = 4-6, and L > 6.]',
     & //,1X,'NOTE:  The L-value is a magnetic coordinate roughly',
     &       ' corresponding to the',
     &  /,1X,'distance in Earth Radii to the',
     &       ' magnetic field line at the magnetic equator.',
     &  /,1x,'For example, a geosynchronous orbit is roughly L = 6.6,',
     &       ' the geographic equator',
     &  /,1x,'is about L = 1, and the heart of the',
     &      ' South Atlantic Anomaly (SAA) is roughly at',
     &  /,1x,'L = 1.2 - 2.',
     &  '  Calculated L-values slightly less than 1 do occur; using',
     &  /,1X,'a lower limit of L = 0 will account for these.')

 452    FORMAT(1X,'The L-values MUST be entered in increasing order',
     &  /,1X,'the L-value of ',F10.2,' has been reset to ',F10.2)

 453    FORMAT(1X,'The number of L-values bins has been reset',
     &  /,1X,' from ',I2,' to ',I2)

 454    FORMAT(/,1X,
     &       'Enter 0 (default) for AP8MIN (1964)',1X,
     &       'or 1 for AP8MAX (1970).')

 455    FORMAT(/,1X,'Enter root name of output TRP files:',
     &  /,1X,'[NOTE:  There will be ',I2,' output files, and',
     &     ' the files for the different L-value'
     &  /,1x,' bins will',
     &       ' be called something.TR# (# = 1,2,...,9,X)]')

 456    FORMAT(1X,'Only 10 L-values are allowed.')

 458    FORMAT(1X,'Calculation reset to whole orbit option, since',
     &         1X,'choosing 1 L bin',
     &       /,1X,' with a minimum L-value equal to 0 is equivalent to',
     &         1X,'the entire orbit.')

 408    FORMAT(/,1X,
     &       'Enter Number of orbits to integrate (default = 200)')

CF? 408    FORMAT(/,1X,
CF?     &        'Enter Number of orbits to integrate (default = 200)',/,
CF?     &        3X,'and Number of steps per orbit')

        END   !TrappedDriverInput routine
```

```
C---------------------------------------------------------------------

        SUBROUTINE OutputTrappedFlux(Evals,NE,TrappedFlux,
     #         DiffTrappedFlux,TrappedFile,OrbIncl,
     #         Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp,
     #         Year,XLbounds,ILbins,PROGRAM_CODE,imod,RelDwellTime,
     #         DistBelt,PreCalcFlux,IPreCalc,ELOWER,EUPPER,MPTS,
     #         Eout,Fluxout,Ndays,Norbits,Norbsteps,
     #         OrbPeriod,OrbPrecTime)

C
C     TEST Routine for writing the orbit-averaged AP8 trapped proton fluxes.
C
      IMPLICIT NONE
      INTEGER*4 I,L,OUTUNIT,IOLEN,NE
      DATA OUTUNIT/2/
      INTEGER*4 NHEADER

      CHARACTER*80 TrappedFile,TEMPFILE
      CHARACTER*9 CREATION_DATE
      CHARACTER*8 CREATION_TIME

      INTEGER iemax,NLvals,ILbins
      PARAMETER (iemax=30,NLvals=10)

      REAL*4 YEAR
      INTEGER*4 VERSION_NUMBER,PROGRAM_CODE,imod
      REAL*4 ELOWER,EUPPER
      CHARACTER*12 TARGET
      DATA TARGET/'UNSHIELDED  '/

      REAL TrappedFlux(iemax,NLvals),Evals(iemax),XLbounds(NLvals)
      REAL RelDwellTime(NLvals)

      REAL DiffTrappedFlux(iemax,NLvals)

      INTEGER MARR,K,MPTS
      PARAMETER (MARR=5000)
      REAL Eout(MARR),Fluxout(MARR,NLvals)


      REAL OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp

      REAL XLinfinite
      PARAMETER (XLinfinite=1.0E+06)

      REAL Fconv


CC     For converting flux to /sr-m**2-s from /cm**2-day, assuming isotropic
CC     flux.  TrappedFlux is in /sr-m**2-s, DiffTrappedFlux in /sr-m**2-s-MeV

CC     Fconv = 1.0E+4/(4*PI*86400).

CC      PARAMETER (Fconv=9.210356E-02)
      PARAMETER (Fconv=7.957747155E+02)  ! 86400 * other Fconv (commented out)

      CHARACTER*4 FEXT(10)

      DATA FEXT/'.TR1','.TR2','.TR3','.TR4','.TR5','.TR6','.TR7',
```

```
     &          '.TR8','.TR9','.TRX'/

      INTEGER IpreCalc
      LOGICAL DistBelt,PreCalcFlux

      INTEGER CREME96_OPEN,stat

      INTEGER Ndays,Norbits,Norbsteps   !passed as arguments, 11-26-97.

C     Add orbital period, 12-1-97.  OrbPrecTime is for future use, 11-26-97.
      REAL Orbperiod,OrbperiodHrs,OrbPrecTime

C     Note that IDistBeltOutput is presently neither used nor output, 11-26-97.

      INTEGER IPreCalcOutput,IDistBeltOutput   !11-26-97.
      INTEGER IPreCalcTmp   !12-1-97, for internal consistency & error checking.

      INTEGER INPUNIT    !11-26-97
      DATA INPUNIT/3/

      REAL PCfluxpts(iemax),DiffPCfluxpts(iemax)     !11-26-97
      INTEGER VERSION_TMP,CODE_TMP      !11-26-97

      REAL PCFluxes(MARR)   !11-26-97

      INTEGER IZLOW,IZHIGH  !11-26-97

      CHARACTER*6 APtitle

      CHARACTER*80 PreCalcFile,TITLELINE
C-----------------------------------------------------------------------------

      CALL GET_CREME96_VERSION(VERSION_NUMBER)
      IPreCalcOutput=0   !11-26-97
      OrbperiodHrs=Orbperiod/3600.0

      IF (PreCalcFlux) THEN
         DistBelt=.FALSE.  !Local variable for header output file.
         IPreCalcOutput=1       !Local variable for header output file.
cf?      IpreCalc = 1 & 3 are AP8MIN calculations, 11-26-97.
cf?         IF ( (IpreCalc .EQ. 1) .OR. (IpreCalc .EQ. 3) )
cf?     &         DistBelt=.TRUE.
      ENDIF

      IDistBeltOutput=0

cf?      IF (DistBelt) IDistBeltOutput=1

      IF (.NOT. PreCalcFlux) THEN  !11-27-97.

C     Open output file and write header

C     ILbins = 0 & ILbins = 1 from input routine are treated
C     as ILbins = 1 for output, since they are stored in the
C     same location in the array.

      IF (ILbins .EQ. 1 .AND. XLBOUNDS(1) .EQ. 0.0) THEN
C        OPEN(UNIT=OUTUNIT,STATUS='NEW',FILE='USER:'//TrappedFile)
        stat = creme96_open(TrappedFile,'user',outunit,'new')
        CALL DATE(CREATION_DATE)
```

```
      CALL TIME(CREATION_TIME)
C
C    Now prepare header for output file:
      NHEADER=23

      WRITE(OUTUNIT,990) NHEADER,TrappedFile(1:70),
     &        VERSION_NUMBER,PROGRAM_CODE

      WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME

      WRITE(OUTUNIT,404) OrbIncl,Apogee,Perigee,AscNodeLong,
     #                AscNodeDisp,PerigDisp


c     RelDwellTime should be 1.0 in this case.

      IF (imod .EQ. 1)
     #  WRITE(OUTUNIT,405) 'AP8MIN',imod-1,IPreCalcOutput,
     #                RelDwellTime(Ilbins),XLbounds(ILbins),
     #                XLbounds(Ilbins+1)

      IF (imod .EQ. 2)
     #  WRITE(OUTUNIT,405) 'AP8MAX',imod-1,IPreCalcOutput,
     #                RelDwellTime(Ilbins),XLbounds(ILbins),
     #                XLbounds(Ilbins+1)

      WRITE(OUTUNIT,411) Norbits,Norbsteps
cf?      WRITE(OUTUNIT,411) Norbits,Norbsteps,OrbPrecTime

      WRITE(OUTUNIT,412) OrbPeriodHrs

      WRITE(OUTUNIT,9195)
      WRITE(OUTUNIT,9196)

      DO I=1,NE,2
         WRITE(OUTUNIT,9200)  (Evals(K),TrappedFlux(K,ILbins)*Fconv,
     #   DiffTrappedFlux(K,ILbins)*Fconv,K=I,I+1)
      END DO

      WRITE(OUTUNIT,9100) ELOWER,EUPPER,MPTS,1,1,TARGET,YEAR,NE,
     #      VERSION_NUMBER,PROGRAM_CODE
      WRITE(OUTUNIT,100)

C     Write trapped proton fluxes to file in standard CREME96 format.

      WRITE(OUTUNIT,200)  (Fconv*FluxOut(K,ILbins),K=1,Mpts)
      WRITE(OUTUNIT,100)

      CLOSE(OUTUNIT)

     ELSE

      DO I=1,66
        IF (TrappedFile(I:I) .NE. ' ' .AND.
     &      TrappedFILE(I+1:I+1) .EQ. ' ') IOLEN=I
      ENDDO

      DO L=1,ILbins
         TEMPFILE=TrappedFILE(1:IOLEN)//FEXT(L)
c         OPEN(UNIT=OUTUNIT,STATUS='NEW',FILE='USER:'//TEMPFILE)
```

```
        stat = creme96_open(TEMPFILE,'user',outunit,'new')
        CALL DATE(CREATION_DATE)
        CALL TIME(CREATION_TIME)

C       Now prepare header for output file:
        NHEADER=23

        WRITE(OUTUNIT,990) NHEADER,TEMPFILE(1:70),
     &        VERSION_NUMBER,PROGRAM_CODE

        WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,
     &        CREATION_TIME

        WRITE(OUTUNIT,404) OrbIncl,Apogee,Perigee,AscNodeLong,
     #                AscNodeDisp,PerigDisp

        IF (L .LT. NLvals) THEN
          IF (imod .EQ. 1)
     #      WRITE(OUTUNIT,405) 'AP8MIN',imod-1,IPreCalcOutput,
     #            RelDwellTime(L),XLbounds(L),XLbounds(L+1)

          IF (imod .EQ. 2)
     #      WRITE(OUTUNIT,405) 'AP8MAX',imod-1,IPreCalcOutput,
     #            RelDwellTime(L),XLbounds(L),XLbounds(L+1)
        ELSE
          IF (imod .EQ. 1)
     #      WRITE(OUTUNIT,405) 'AP8MIN',imod-1,IPreCalcOutput,
     #            RelDwellTime(L),XLbounds(L),XLinfinite
          IF (imod .EQ. 2)
     #      WRITE(OUTUNIT,405) 'AP8MAX',imod-1,IPreCalcOutput,
     #            RelDwellTime(L),XLbounds(L),XLinfinite
        ENDIF

      WRITE(OUTUNIT,411) Norbits,Norbsteps
cf?      WRITE(OUTUNIT,411) Norbits,Norbsteps,OrbPrecTime

     WRITE(OUTUNIT,412) OrbPeriodHrs

      WRITE(OUTUNIT,9195)
      WRITE(OUTUNIT,9196)

      DO I=1,NE,2
        WRITE(OUTUNIT,9200) (Evals(K),TrappedFlux(K,L)*Fconv,
     #  DiffTrappedFlux(K,L)*Fconv,K=I,I+1)
      END DO

     WRITE(OUTUNIT,9100) ELOWER,EUPPER,MPTS,1,1,TARGET,YEAR,NE,
     #     VERSION_NUMBER,PROGRAM_CODE
      WRITE(OUTUNIT,100)

C     Write trapped proton fluxes to file in standard CREME96 format.

      WRITE(OUTUNIT,200) (Fconv*FluxOut(K,L),K=1,Mpts)
      WRITE(OUTUNIT,100)

      CLOSE(OUTUNIT)

     ENDDO       !number of L-bins.

    ENDIF   !choosing between whole orbit and L-bin options.
```

```
      ELSE   !handle pre-calculated trapped fluxes, 11-26-97.

C       Presently, there are no L-bins for pre-calculated trapped fluxes.

        IF (IPreCalc .EQ. 0) PreCalcFile='IPREC0.TRP'
        IF (IPreCalc .EQ. 1) PreCalcFile='IPREC1.TRP'
        IF (IPreCalc .EQ. 2) PreCalcFile='IPREC2.TRP'
        IF (IPreCalc .EQ. 3) PreCalcFile='IPREC3.TRP'

        stat = creme96_open(PreCalcFile,'cr96tables',inpunit,'old')
        stat = creme96_open(TrappedFile,'user',outunit,'new')

        CALL DATE(CREATION_DATE)
        CALL TIME(CREATION_TIME)

        NHEADER=23

        WRITE(OUTUNIT,990) NHEADER,TrappedFile(1:70),
     &       VERSION_NUMBER,PROGRAM_CODE

        WRITE(OUTUNIT,992) VERSION_NUMBER,CREATION_DATE,CREATION_TIME

        READ(INPUNIT,9194)TITLELINE   !dummy title lines from creation of
        READ(INPUNIT,9194)TITLELINE   !pre-calculated trapped fluxes.

        READ(INPUNIT,1404) OrbIncl,Apogee,Perigee,AscNodeLong,
     #                AscNodeDisp,PerigDisp

        WRITE(OUTUNIT,404) OrbIncl,Apogee,Perigee,AscNodeLong,
     #                AscNodeDisp,PerigDisp

        READ(INPUNIT,1405) APtitle,imod,IPreCalcTmp

        READ(INPUNIT,1406)RelDwellTime(Ilbins),XLbounds(ILbins),
     #                XLbounds(Ilbins+1)

        WRITE(OUTUNIT,405) APtitle,imod,IPreCalcOutput,
     #                RelDwellTime(Ilbins),XLbounds(ILbins),
     #                XLbounds(Ilbins+1)

        READ(INPUNIT,1411) Norbits,Norbsteps
        WRITE(OUTUNIT,411) Norbits,Norbsteps

        READ(INPUNIT,1412) OrbPeriodHrs
        WRITE(OUTUNIT,412) OrbPeriodHrs

        READ(INPUNIT,9194)TITLELINE
        WRITE(OUTUNIT,9194)TITLELINE
        READ(INPUNIT,9194)TITLELINE
        WRITE(OUTUNIT,9194)TITLELINE

        NE = 29  !presently hardwired for pre-calculated fluxes.

        DO I=1,NE,2
            READ(INPUNIT,9201) Evals(I),PCfluxpts(I),
     #      DiffPCfluxpts(I),Evals(I+1),PCfluxpts(I+1),
     #      DiffPCfluxpts(I+1)
            WRITE(OUTUNIT,9200) (Evals(K),PCfluxpts(K),
     #      DiffPCfluxpts(K),K=I,I+1)
```

```
      END DO

      READ(INPUNIT,9100) ELOWER,EUPPER,MPTS,IZLOW,IZHIGH,TARGET,
     #       YEAR,NE,VERSION_TMP,CODE_TMP
      READ(INPUNIT,100)

      WRITE(OUTUNIT,9100) ELOWER,EUPPER,MPTS,IZLOW,IZHIGH,TARGET,
     #       YEAR,NE,VERSION_NUMBER,PROGRAM_CODE
      WRITE(OUTUNIT,100)

C     Write trapped proton fluxes to file in standard CREME96 format.

      READ(INPUNIT,200) (PCFluxes(K),K=1,Mpts)
      WRITE(OUTUNIT,200) (PCFluxes(K),K=1,Mpts)

      READ(INPUNIT,100)
      WRITE(OUTUNIT,100)

      CLOSE(INPUNIT)
      CLOSE(OUTUNIT)

     ENDIF  !for regular vs. pre-calculated trapped fluxes, 11-26-97.


C    FORMAT statements

100  FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,
    &       2X,0PF8.3,1x,I2,1x,I1,1x,I4,1x,I1)

200  FORMAT((1X,6(1PE10.4,2X)))

 404 FORMAT(1x,'%Incl = ',F7.3,' deg  Apo = ',E10.4,
    #          ' Peri = ',E10.4,' km',1x,3(F6.2,1x))

 405 FORMAT(1x,'%',A6,1X,'IMOD =',I2,1X,'IPRECALC =',I2,/,1X,
    #          '%Relative dwell time = ',E10.4,1X,
    #          'L Bin: ',2(E10.4,1X))

C    Number of steps per orbits is presently fixed at 200, and
C    the orbital procession period is not presently calculated, 11-26-97.

 411 FORMAT(1x,'%No. orbits =',I8,2X,'No. steps/orbit = ',I6)

cf? 411  FORMAT(1x,'%No. orbits =',I8,2X,'No. steps/orbit = ',I6,
cf?     #          'Precession Period = ',E10.4)

 412 FORMAT(1x,'%Orbital period = ',F8.2,1X,'hours')

 990 FORMAT(I3,1x,A70,I4,1x,I1)
 992 FORMAT(1x,'%Created by TRAPPED_DRIVER Version ',I4,
    &          ' on ',A9,' at ',A8)

9100 FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,2X,0PF8.3,1X,I4,1X,I4,1X,I1)
9195 FORMAT(1X,'%Calculated energies, integral fluxes, and ',
    &         'differential fluxes')
9196 FORMAT(1X,'%',3X,'MeV',7X,'/m**2-sr-s',2X,'/m**2-sr-s-MeV',3X,
    &         'MeV',7X,'/m**2-sr-s',2X,'/m**2-sr-s-MeV')
9200 FORMAT((1X,'%',1X,3(1PE10.4,2X),3X,3(1PE10.4,2X)))
9210 FORMAT((1X,3(1PE10.4,2X)))
```

```
C       FORMAT lines for reading pre-calculated files, 11-26-97 and 12-1-97.

1404    FORMAT(1x,8X,F7.3,12X,E10.4,8X,E10.4,3X,1x,3(F6.2,1x))

1405    FORMAT(1x,1X,A6,1X,6X,I2,1X,10X,I2)
1406    FORMAT(1X,23X,E10.4,1X,7X,2(E10.4,1X))

1411    FORMAT(1x,13X,I8,2X,18X,I6)
1412    FORMAT(1x,18X,F8.2,1X,5X)

9194    FORMAT(1X,A79)
9201    FORMAT((1X,1X,1X,3(1PE10.4,2X),3X,3(1PE10.4,2X)))

        RETURN
        END

C-----------------------------------------------------------------------

           SUBROUTINE Trapped_ORBINT(OrbIncl,Apogee,Perigee,AscNodeLong,
     #              AscNodeDisp,PerigDisp,Evals,NE,sumexp,d,
     #              Year,XLbounds,ILbins,imod,RelDwellTime,
     #              DistBelt,PreCalcFlux,IPreCalc,Ndays,
     #              Norbits,Norbsteps,Period,OrbPrecTime)

           IMPLICIT NONE

           INTEGER J,Jmax,L,Ndays,NorbSteps,IPreCalc,NLvals,Norbits
           INTEGER NE,imod

C          Establish Norbits & Norbsteps in TrappedDriverInput, 11-26-97.
           PARAMETER (NLvals=10)

           LOGICAL DistBelt,PreCalcFlux

C          Initial Orbital parameters are set in Subroutine TrappedDriverInput.

           REAL OrbIncl,Apogee,Perigee,AscNodeLong,AscNodeDisp,PerigDisp
           REAL Time,Period,Step

C          Parameters along each orbital step
           REAL Zlat,Zlon,Alt

           INTEGER ILbins,ILbin,ICODE,NperLbin(NLvals)
           REAL Year,XLval,BB0,XLbounds(NLvals),XLinfinite
           PARAMETER (XLinfinite=1.0E+06)

           INTEGER IE,iemax
           parameter (iemax = 30)   ! MAX NO. OF ENERGIES ALLOWED
           real Evals(iemax)

           REAL RelDwellTime(NLvals)

           REAL flux(iemax), expose(iemax)
           REAL sumexp(iemax,NLvals), fluxi(iemax,NLvals)
           REAL difed1(iemax,NLvals), d(iemax,NLvals)

           real*4  b, delt  !B is mag. field, delt=step (time interval)
           real*4  yearp

         real*8 gmagmo
```

```
      common /gmagmo/ gmagmo!     for esa trara1n

      SAVE

      real OrbPrecTime   !for future use, 12-1-97.

C-----------------------------------------------------------------------

C        Initializations

         DO L=1,NLvals
          NperLbin(L)=0
          do  ie=1,ne    ! initailize arrays
             sumexp(ie,L) = 0.
             fluxi(ie,L) = 0.
          enddo
         ENDDO

          do  ie=1,ne    ! initailize arrays
             expose(ie) = 0.0
          enddo

         IF (PreCalcFlux) THEN


C            The output routine OutputTrappedFlux handles getting the
C            pre-calculated trapped fluxes.  We simply return with
C            the proper flag.

           RETURN  !could just use subsequent RETURN, since this IF statement
                   !skips all lines before the subsequent RETURN

         ELSE  !calculate trapped flux if not using pre-calculated ones

C          Initialize Orbit routine

           CALL Orbit(1,Period,ZLon,ZLat,Alt,Apogee,Perigee,OrbIncl,
     #                   AscNodeLong,AscNodeDisp,PerigDisp)

C          Compute the total number of steps in "Ndays" days if we make
C          "Norbsteps"  steps per orbit.  Use 2 days and 200 steps per orbit
C          presently.
C

           NDAYS=INT(Norbits*PERIOD/86400. + 1.0)
           JMAX=INT(Ndays* NorbSteps*86400./PERIOD + 1.5)

C
C          Compute the step size in seconds.
C
           STEP=PERIOD/FLOAT(NorbSteps)

           delt=STEP

C
          DO J=1,JMAX

           time=FLOAT(j-1)*step

           CALL Orbit(2,Time,ZLon,ZLat,Alt,Apogee,Perigee,OrbIncl,
```

```
#                      AscNodeLong,AscNodeDisp,PerigDisp)

          call blccoords(Zlat,Zlon,Alt,Year,imod,BB0,XLval,yearp,B)

         IF ( XLbounds(2) .LT. XLinfinite .OR.
     &        ( XLbounds(2) .GE. XLinfinite .AND.
     &          XLbounds(1) .GT. 0.0 ) ) THEN

          IF (XLval .GT. 99999.0) XLval=99999.0

          CALL GetLbin(XLval,XLbounds,ILbins,ILbin)

          IF (ILbin .GE. 1 .AND. ILbin .LE. NLvals)
     &        NperLbin(ILbin)=NperLbin(ILbin)+1

         ELSE

C         If no L-bins are specified or 1 L-bin is specified
C         and the lower bound is L = 0, use only the first
C         element of the array.  In this case, the following
C         sum should equal JMAX once the stepping through the
C         orbit is completed.

          ILbin=1
          NperLbin(Ilbin)=NperLbin(Ilbin)+1

         ENDIF

c       Now pass B, L (XLval), imod, e , ne to subroutine trapped_protons,
c       return integral flux and yearp

        call trapped_protons(B,XLval,yearp,imod,evals(1),flux(1),ne)

         IF (ILbin .GE. 1 .AND. ILbin .LE. NLvals) THEN
          do ie = 1, ne
             expose(ie) = flux(ie) * delt
             sumexp(ie,ilbin) = sumexp(ie,ilbin) + expose(ie)
          enddo
         ENDIF !for within allowed Lbins

        ENDDO    !for number of orbital steps (up to JMAX)

        DO L=1,ILbins
          do ie = 1, ne
             sumexp(ie,L) = sumexp(ie,L)/(FLOAT(NperLbin(L))*delt)
          enddo

          RelDwellTime(L) =    FLOAT(NperLbin(L))/
     &                             FLOAT(Jmax)

          do  ie = 1,ne-1
             difed1(ie,L) = sumexp(ie,L) - sumexp(ie+1,L)
          enddo

          difed1(ne,L) = sumexp(ne,L)

        ENDDO  !stepping through L-bins

        call differ(ne, evals, sumexp, difed1, d, Ilbins)
```

```
       do L = 1,ILbins
         do ie = 1, ne
           IF (d(ie,L) .LE. 1.0E-20) d(ie,L) = 0.0
         enddo
       enddo


      ENDIF   !for using either pre-calculated or directly-calculated Fluxes

      RETURN
      END

C-------------------------------------------------------------------------------

C     The following GetLbin routine  is identical to the routine in GEOMAG96.
C     In order to avoid linking with all of GEOMAG96, it is included here
C     for now.  Before releasing, this should be in a standalone module
C     that can be called from either the GTRANS_DRIVER or the TRAPPED_DRIVER.

C-------------------------------------------------------------------------------

      SUBROUTINE GetLbin(XLval,XLbounds,ILbins,ILbin)

      IMPLICIT NONE
      INTEGER ILbins,ILbin,NLvals,L
      PARAMETER (NLvals=10)
      REAL XLval,XLbounds(NLvals)
      LOGICAL FindLbin

C     No attempt is made to eliminate "unphysical" or "approximate"
C     L-values using the ICODE returned from GET_BLCOORDS, since any
C     analyses using L-values are likely to handle these locations
C     "as is", i.e. with the calculated L-value.

C-------------------------------------------------------------------------------

      FindLbin=.TRUE.

      ILbin=0

      DO L=1,ILbins
       IF (FindLbin) THEN

         IF (L .LT. NLvals) THEN
           IF ( (XLval .GE. XLbounds(L)) .AND.
     &     (XLval .LT. XLbounds(L+1)) )THEN
             ILbin=L
             FindLbin=.FALSE.
           ENDIF

         ELSE  !special handling of L=NLvals case

           IF (XLval .GE. XLbounds(L))THEN
             ILbin=L
             FindLbin=.FALSE.
           ENDIF
         ENDIF   !checking of each L-bin

       ENDIF     !for FINDLbin logical
      ENDDO
```

```
      RETURN
      END

C-----------------------------------------------------------------------
C     This should be a dead subroutine now, 11-26-97.

      SUBROUTINE GetPreCalcFlux(IPreCalc,Evals,TrappedFlux,
     &             DiffTrappedFlux,DistBelt,imod)

      IMPLICIT NONE

      INTEGER IPreCalc,NLvals,NE,imod,I,L,iemax

      PARAMETER (iemax=30,NLvals=10)

      LOGICAL DistBelt

C     Initial Orbital parameters are set in Subroutine TrappedDriverInput.

      real Evals(iemax)

      REAL Trappedflux(iemax,NLvals),DiffTrappedflux(iemax,NLvals)
      REAL RelDwellTime(NLvals)

C-----------------------------------------------------------------------
      RelDwellTime(1)=1.0

      DO I=1,iemax
        DO L=2,NLvals
          Trappedflux(i,L)=0.0
          DiffTrappedflux(i,L)=0.0
        ENDDO
      ENDDO

      DO L=2,NLvals
        RelDwellTime(L)=0.0
      ENDDO

C     Need to set Evals, Trappedflux(*,1), & DiffTrappedflux(*,1)
C     when actually implement routine.  Also change STOP TO RETURN

      WRITE(*,'(1X,''Pre-calculated trapped fluxes are presently'',
     # 1X,''not available.'',/,1X,''Aborting Trapped_Driver'')')

      STOP   !       RETURN
!      RETURN

      END

C-----------------------------------------------------------------------

      SUBROUTINE Trapped_Spectra(Evals,NE,ELOWER,EUPPER,M,ILbins,
     #             TrappedFlux,DiffTrappedFlux,Eout,Fluxout)

      IMPLICIT NONE

      INTEGER NLvals,IEmax,NE,ILbins
      PARAMETER (NLvals=10,IEmax=30)
```

```
      REAL Evals(IEmax),DiffTrappedFlux(IEmax,NLvals)
      REAL TrappedFlux(IEmax,NLvals)
      REAL DiffFluxSPin(IEmax)

      REAL ELOWER,EUPPER,DE

      INTEGER MARR,M,I,J,K,NEsp,K1
      PARAMETER (MARR=5000)
      REAL Eout(MARR),Fluxout(MARR,NLvals),DiffFluxSPout(MARR)

      REAL D2FLUX(500),DUMMYFLUX(500),DERIVLOW,DERIVHIGH

      LOGICAL NATURAL(2)
      INTEGER NElogmin,NElogmax,NElog

      REAL D2LOGFLUX(500),DUMMYLOGFLUX(500)
      REAL DERIVLOGLOW,DERIVLOGHIGH
      REAL EvalsLog(IEmax),Elogout(MARR),DiffFluxLogout(MARR)
      REAL DiffFluxlog(IEmax)

      INTEGER Jmax     !11-24-97

C-----------------------------------------------------------------------------

C     Compute energies on logaritmically-spaced grid

      M=1002

      DE=(EUPPER/ELOWER)**(1./(M-1.))
      EOUT(1)=ELOWER
      ELOGOUT(1)=LOG(EOUT(1))

      DO J=2,M-1
        EOUT(J)=EOUT(J-1)*DE
      ELOGOUT(J)=LOG(EOUT(J))
      END DO

      EOUT(M)=EUPPER
      ELOGOUT(M)=LOG(EOUT(M))

C     Initialize FluxOut
      DO I = 1, ILbins
        DO J=1,M
          FluxOut(J,I)=0.0
        END DO
      END DO

      EOUT(M)=EUPPER

      NATURAL(1)=.TRUE.
      NATURAL(2)=.TRUE.

      DO I = 1, ILbins   !set up and CALL SPLINE for each L-value bin.

        NEsp=0
        NElogmin=0
        NElogmax=0
        Jmax=0

        DO J=1,IEMAX   !establish array for passing into SPLINE & SPLINT
```

```
          DiffFluxSPin(J)=DiffTrappedFlux(J,I)
c           Eliminate points at which differential flux is not well
C           behaved (monotonically decreasing), 11-24-97.
          IF (DiffFluxSPin(J) .GT. 0.0) THEN

c           Eliminate points at which differential flux=0, 11-24-97.

            Jmax=J
            NEsp=NEsp+1
            IF (NElogmin .EQ. 0) NElogmin=J
            DiffFluxLog(NEsp-Nelogmin+1)=LOG(DiffFluxSPin(J))
            EvalsLog(NEsp-Nelogmin+1)=LOG(Evals(J))   !J to NEsp, 11-24-97.
          ENDIF
        ENDDO
        NElogmax=NEsp

        NElog=NElogmax-NElogmin+1

        IF (Nesp .GT. 1) THEN

          CALL SPLINE(EvalsLog,DiffFluxLog,NElog,500,NATURAL
     >                ,0.0,0.0,D2LOGFLUX,DUMMYLOGFLUX)

          DO K=1,M       !for calculating at the standard CREME96 energies.

            CALL SPLINT(Evalslog,DiffFluxLog,D2LOGFLUX,
     >                  NElog,Elogout(K),DiffFluxLogout(K))

            IF (Eout(K) .LT. 1.0 .OR.
     &          Eout(K) .GT. Evals(Jmax)) THEN   !NEsp to J, 11-24-97.
              Fluxout(K,I)=0.0
            ELSE
              Fluxout(K,I)=EXP(DiffFluxlogout(K))
            ENDIF

          ENDDO          !K=1,M

        ENDIF

      ENDDO   !I=1,ILbins


      RETURN
      END
```

```
      SUBROUTINE ULET96(SLOWER,SUPPER,TARGET,
     &                  ELOWER,EUPPER,M,IZLO,IZUP,
     &                  FLUX,LDUM,
     &                  SPECT)
C
C Creates an LET spectrum in standard format from an energy
C spectrum in standard format.
C
C Inputs:
C SLOWER,SUPPER = min, max LET values (in MeV-cm2/g -- not /mg!)
C TARGET        = target material (CHARACTER*7, generally SILICON)
C FLUX    Contains the energy spectra of up to NELM elements specified at
C         up to MARR energies
C ELOWER,EUPPER = min, max energy of particle spectra
C M             = number of bins in energy spectra
C IZLO,IZUP     = min, max atomic numbers in LET spectrum.
C LDUM          = number of bins in integral LET spectrum (</= LARR=1002)
C Outputs:
C SPECT         = output LET spectrum
C
C***********************************************************************
      IMPLICIT NONE
      INTEGER*4 MARR,NELM,LARR,M,IZLO,IZUP,LDUM,L,J,K,I,IK
      PARAMETER (MARR=5000,NELM=92,LARR=1002)
      REAL*4 FLUX(NELM,MARR),E(MARR),SP(NELM,MARR)
      REAL*4 SPECT(LARR),SL(LARR)
      REAL*4 AMASS
      COMMON/MASS/AMASS(109)
      CHARACTER TARGET*12
      REAL*4 SLOWER,SUPPER,ELOWER,EUPPER,DE,DS,RSP,FUN,XK,ADD

C     Construct list of energies

      DE=(EUPPER/ELOWER)**(1./(M-1.))
      E(1)=ELOWER
      DO J=2,M-1
        E(J)=E(J-1)*DE
      END DO
      E(M)=EUPPER

      L=LDUM
      IF (L.GT.LARR) L=LARR

C     Construct list of stopping powers

      DS=(SUPPER/SLOWER)**(1./(L-1.))
      SL(1)=SLOWER
      DO J=2,L
        SL(J)=SL(J-1)*DS
      END DO
      SL(L)=SUPPER

C     Now get table of stopping powers
      CALL UNLOAD_STABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,SP)

C     Initialize spectrum

      DO J=1,L
        SPECT(J)=0.
      END DO
```

```
C     For each energy find stopping power index and increment
C     all lower stopping powers

      RSP=1./SLOWER
      FUN=1./LOG(SUPPER/SLOWER)
      DO J=IZLO,IZUP
        DO K=1,M
          XK=1.+(L-1.)*LOG(SP(J,K)*RSP)*FUN
          IK=INT(XK)
          IF (K.EQ.1) THEN
            ADD=FLUX(J,K)*(E(K+1)-E(K))*0.5
          ELSE IF (K.EQ.M) THEN
            ADD=FLUX(J,K)*(E(K)-E(K-1))*0.5
          ELSE
            ADD=FLUX(J,K)*(E(K+1)-E(K-1))*0.5
          ENDIF
          DO I=1,IK
            IF (I.LT.IK) THEN
              SPECT(I)=SPECT(I)+ADD
            ELSE IF (I.EQ.IK) THEN
              SPECT(I)=SPECT(I)+ADD*(XK-IK)
            ENDIF
          END DO
        END DO
      END DO

      RETURN
      END
```

```
      SUBROUTINE UNLOAD_CREME96_FLUX(INFILE,
     &                                 ELOWER,EUPPER,M,IZLO,IZUP,
     *                                 FLUX)

      IMPLICIT NONE
      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 FLUX(NELM,MARR)
      REAL*4 ELOWER,EUPPER
      INTEGER*4 M,IZLO,IZUP,J,K,IVER,KVER,KPROG,NHEADER
      INTEGER*4 STAT,CREME96_OPEN
      CHARACTER*80 INFILE,ILINE

      CALL CHECK_CREME96_VERSION(INFILE,IVER)

c      OPEN(UNIT=25,STATUS='OLD',READONLY,FILE='USER:'//INFILE)
      stat = creme96_open(infile,'user',25,'old')

      IF (IVER.GE.102) THEN
      READ(25,*) NHEADER
      DO J=1,NHEADER
         READ(25,110) ILINE
 110  FORMAT(A80)
      ENDDO
      ENDIF

      READ(25,*) ELOWER,EUPPER,M,IZLO,IZUP
      READ(25,*)
      DO 100 J=IZLO,IZUP
        READ(25,*) (FLUX(J,K),K=1,M)
        READ(25,*)
C         WRITE(6,999) J,(FLUX(J,K),K=1,6)
C         WRITE(6,999) J,(FLUX(J,K),K=997,1002)
 999    FORMAT(1X,I3,6E11.4)
 100  CONTINUE

      CLOSE(UNIT=25)
      RETURN
      END
```

```
      SUBROUTINE UNLOAD_CTABLE(ELOWER,EUPPER,N,NSP,IZLO,IZUP,TARGET,
     &                          CC,SPLOSS)
C
C     Subroutine used by UPROPC to unload cross-section tables.

      IMPLICIT NONE
      INTEGER*4 MARR,NELM,MCS,STAT,CREME96_OPEN
      PARAMETER (MARR=5000,NELM=92,MCS=10)
      REAL*4 CC(NELM,NELM,MCS)
      REAL*4 SPLOSS(NELM,MCS)
      REAL*4 ELOWER,EUPPER,ELOWER$,EUPPER$
      INTEGER*4 N,NSP,NABS,IZLO,IZUP,N$,NSP$,IZLO$,IZUP$,I,J,K
      CHARACTER*12 TARGET,TARGET$
      CHARACTER*80 CTABLEFILE,SPTABLEFILE

      DATA ELOWER$,EUPPER$,N$,IZLO$,IZUP$,
     &      TARGET$/0.,0.,0,0,0,' '/

C     FORMAT Statements

100   FORMAT(1X,2(1PE10.4,2X),4(I5,2X),A12,2X,1PE10.4)

      NABS=ABS(N)
C
C     First, check standard table:

c      CTABLEFILE='CREME96:CTABLE.STD'
c      SPTABLEFILE='CREME96:SPTABLE.STD'
c      IF (IZLO.GT.28 .or. IZUP.GT.28) THEN
c          CTABLEFILE='CREME96:CTABLE.XTD'
c          SPTABLEFILE='CREME96:SPTABLE.XTD'
c      ENDIF
      CTABLEFILE='CTABLE.STD'
      SPTABLEFILE='SPTABLE.STD'
      IF (IZLO.GT.28 .or. IZUP.GT.28) THEN
          CTABLEFILE='CTABLE.XTD'
          SPTABLEFILE='SPTABLE.XTD'
      ENDIF

c      OPEN(UNIT=36,STATUS='OLD',FILE=CTABLEFILE,READONLY,SHARED)
      stat = creme96_open(ctablefile,'cr96tables',36,'old')
      READ(36,100) ELOWER$,EUPPER$,N$,IZLO$,IZUP$,NSP$,TARGET$
CC        WRITE(6,100) ELOWER$,EUPPER$,N$,IZLO$,IZUP$,NSP$,TARGET$

      IF (ELOWER.EQ.ELOWER$ .AND. EUPPER.EQ.EUPPER$ .AND.
     $     NABS.EQ.N$ .AND. NSP.EQ.NSP$ .AND.
     &     TARGET.EQ.TARGET$ .AND.
     &    (IZLO$.LE.IZLO .AND. IZLO.LE.IZUP$) .AND.
     &    (IZLO$.LE.IZUP .AND. IZUP.LE.IZUP$)) THEN

C     Standard table contains the necessary information.
c      OPEN(UNIT=37,STATUS='OLD',FILE=SPTABLEFILE,READONLY,SHARED)
      stat = creme96_open(sptablefile,'cr96tables',37,'old')
      READ(37,100)
CC        READ(37,100) ELOWER$,EUPPER$,N$,IZLO$,IZUP$,NSP$,TARGET$
CC        WRITE(6,100) ELOWER$,EUPPER$,N$,IZLO$,IZUP$,NSP$,TARGET$
C      WRITE(6,999) CTABLEFILE(1:20),SPTABLEFILE(1:20),TARGET
 999  FORMAT(' Standard tables ',A20,1x,A20,
     &     /,' of nuclear cross-sections',
     &       ' in ',A12,' used for transport calculation.')
```

```
      GOTO 500
      ENDIF

C     Check if appropriate tables exist in USER area:

      CLOSE(36)
c      OPEN(UNIT=36,STATUS='OLD',FILE='USER:CTABLE.DAT',READONLY,ERR=50)
      stat = creme96_open('ctable.dat','user',36,'old')
      if (stat .ne. 0) goto 50
      READ(36,100) ELOWER$,EUPPER$,N$,IZLO$,IZUP$,NSP$,TARGET$

      IF (ELOWER.EQ.ELOWER$ .AND. EUPPER.EQ.EUPPER$ .AND.
     $    NABS.EQ.N$ .AND. NSP.EQ.NSP$ .AND.
     &    TARGET.EQ.TARGET$ .AND.
     &   (IZLO$.LE.IZLO .AND. IZLO.LE.IZUP$) .AND.
     &   (IZLO$.LE.IZUP .AND. IZUP.LE.IZUP$)) THEN

C     Standard table contains the necessary information.
c      OPEN(UNIT=37,STATUS='OLD',FILE='USER:SPTABLE.DAT',READONLY,ERR=50)
      stat = creme96_open('sptable.dat','user',37,'old')
      if (stat .ne. 0) goto 50
      READ(37,100)
      WRITE(6,998) TARGET
 998  FORMAT(' User tables (USER:CTABLE.DAT & SPTABLE.DAT) of'
     &       ' nuclear cross-sections'
     &     /,' in ',A12,' used for transport calculation.')
      GOTO 500


      ELSE
 50   CONTINUE
C
      CLOSE(36)
      CLOSE(37)
      WRITE(6,997)
 997  FORMAT(' Non-standard energy or shielding'
     &       ' in transport calculation.',
     &     /,' Create new cross-section tables in USER area.')

      CALL CTABLE(ELOWER,EUPPER,NABS,NSP,IZLO,IZUP,TARGET)
      READ(36,100) EUPPER$,ELOWER$,N$,IZLO$,IZUP$,NSP$,TARGET$
c      OPEN(UNIT=37,STATUS='OLD',FILE='USER:SPTABLE.DAT',READONLY)
      stat = creme96_open('sptable.dat','user',37,'old')
      READ(37,100)

      ENDIF

 500  CONTINUE
      READ(36,100)
      READ(37,100)
        DO J=1,NABS
          READ(36,*) ((CC(K,I,J),K=IZLO$,IZUP$),I=IZLO$,IZUP$)
          READ(37,*) (SPLOSS(K,J),K=IZLO$,IZUP$)
        END DO
      CLOSE(UNIT=36)
      CLOSE(UNIT=37)
      RETURN
      END
```

```
      SUBROUTINE UNLOAD_HEADERS(INFILE,NHMAX,HEADER_LINE,LINEMAX)
C
C     Reads lines of header information from file INFILE and
C     returns LINEMAX lines in array HEADER_LINE.  Maximum number
C     of returned lines set by input NHMAX.

      IMPLICIT NONE
      CHARACTER*80 INFILE,HEADER_LINE
      DIMENSION HEADER_LINE(1)
      INTEGER*4 NHMAX,NHEADERS,VERSION_NUMBER,LINEMAX
      INTEGER*4 STAT,CREME96_OPEN
      INTEGER*4 INUNIT,J
      DATA INUNIT/4/

      CALL CHECK_CREME96_VERSION(INFILE,VERSION_NUMBER)
      CALL CHECK_HEADER_LENGTH(INFILE,NHEADERS)

c      OPEN(UNIT=INUNIT,FILE='USER:'//INFILE,
c     &     STATUS='OLD',READONLY,SHARED)
      stat = creme96_open(infile,'user',inunit,'old')

C     By pass first line:
      IF (VERSION_NUMBER.GE.102) READ(INUNIT,999)
C     Now store headers:
      LINEMAX=MIN(NHMAX,NHEADERS)
      DO J=1,LINEMAX
      READ(INUNIT,999) HEADER_LINE(J)
  999 FORMAT(A80)
      ENDDO

      CLOSE(INUNIT)

      RETURN
      END
```

```
      SUBROUTINE UNLOAD_LET_SPECTRUM(LETFILE,XL,FLUX,NPTS)
C
C     Subroutine to unload integral LET Spectrum from input file into
C     array.  Can handle either CREME96 format or the old CREME format
C     (ie., two-column table of LET (in MeV-cm2/g) and particle fluxes
C     (in particles/m2/s/sr).)  CREME96 format is denoted by the
C     suffix .LET or .DLT in the filename.
C
C     Written by:     Allan J. Tylka
C                     Code 7654
C                     Naval Research Laboratory
C                     Washington, DC 20375-5352
C                     tylka@crs2.nrl.navy.mil
C
C     Last update:    31 October 1996: modified to read differential LET files.
C
C-----------------------------------------------------------------------------
C
      IMPLICIT NONE
      INTEGER*4 K,N,NZ,NZT,I,NPTS,ILONG,MAXSPEC,STAT,CREME96_OPEN
      INTEGER*4 IVER,J,NHEADER
      PARAMETER (MAXSPEC=5000)
      CHARACTER*80 LETFILE,ILINE
      REAL*4 XL,FLUX,DUMFLUX,DUMXL,EL,EU
      DIMENSION XL(1),FLUX(1)
      DIMENSION DUMFLUX(MAXSPEC),DUMXL(MAXSPEC)

      DO 1 I=1,MAXSPEC
         DUMFLUX(I)=0.0
 1    CONTINUE

      CALL CHECK_CREME96_VERSION(LETFILE,IVER)

C      OPEN(UNIT=10,FILE='USER:'//LETFILE,STATUS='OLD',READONLY)
      stat = creme96_open(letfile,'user',10,'old')
      ILONG=INDEX(LETFILE,'.')


      IF (LETFILE(ILONG+1:ILONG+3).EQ.'LET' .or.
     &    LETFILE(ILONG+1:ILONG+3).EQ.'let' .or.
     &    LETFILE(ILONG+1:ILONG+3).EQ.'DLT' .or.
     &    LETFILE(ILONG+1:ILONG+3).EQ.'dlt') THEN

        IF (IVER.GE.102) THEN
        READ(10,*) NHEADER
        DO J=1,NHEADER
           READ(10,110) ILINE
 110       FORMAT(A80)
        ENDDO
        ENDIF

        read(10,*) el,eu,n,nz,nzt
 200    FORMAT((1X,6(1PE10.4,2X)))

C       Calculate abscissae (LET values)

        DUMXL(1)=el
        DUMXL(N)=eu
        do 100 i=2,N-1
        DUMXL(i)=el*(eu/el)**(float(i-1)/float(n-1))
```

```
100        continue

C          Read blank line
           read(10,110) ILINE

c          read in the integral LET spectrum
           read(10,200) (dumflux(i),i=1,n)
           CLOSE(10)

        ELSE

C          Two-column format  (not CREME96 standard)
           N=1
 10        CONTINUE
           READ(10,*,END=15) DUMXL(N),DUMFLUX(N)
           N=N+1
           GOTO 10
 15        CONTINUE
           N=N-1
           CLOSE(10)

        ENDIF

C
C       Editing of input LET spectrum removed by AJT 10-21-96.
C
        NPTS=0
        DO 1000 K=1,N
                   NPTS=NPTS+1
                   XL(NPTS)=DUMXL(K)
                   FLUX(NPTS)=DUMFLUX(K)
 1000   CONTINUE

        RETURN
        end
```

```
      SUBROUTINE UNLOAD_PARTIAL_FLUX(INFILE,IZMIN,IZMAX,EMINCUT,EMAXCUT,
     &                                ELOWER,EUPPER,M,IZLO,IZUP,
     *                                FLUX)
C
C     From specified file INFILE unloads only elements IZMIN le Z le IZMAX
C     and returns the spectra in array FLUX.
C

      IMPLICIT NONE
      INTEGER*4 MARR,NELM,STAT,CREME96_OPEN
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 FLUX(NELM,MARR),FLUXDUM(NELM,MARR),E(MARR)
      REAL*4 ELOWER,EUPPER,EMINCUT,EMAXCUT,DE
      INTEGER*4 IZMIN,IZMAX,M,IZLO,IZUP,KZLO,KZUP,J,K
      INTEGER*4 KMIN,KMAX,KMIN1,KMAX1
      INTEGER*4 IVER,NHEADER
      CHARACTER*80 INFILE,ILINE


      CALL CHECK_CREME96_VERSION(INFILE,IVER)

c       OPEN(UNIT=25,STATUS='OLD',READONLY,FILE='USER:'//INFILE)
      stat = creme96_open(infile,'user',25,'old')

      IF (IVER.GE.102) THEN
      READ(25,*) NHEADER
      DO J=1,NHEADER
         READ(25,110) ILINE
 110  FORMAT(A80)
      ENDDO
      ENDIF

      READ(25,*) ELOWER,EUPPER,M,KZLO,KZUP
      READ(25,*)
      DO 100 J=KZLO,KZUP
        READ(25,*) (FLUXDUM(J,K),K=1,M)
        READ(25,*)
 100  CONTINUE
      CLOSE(UNIT=25)

      IF (IZMIN.EQ.0) THEN
           IZLO=KZLO
      ELSE
           IZLO=MAX(IZMIN,KZLO)
      ENDIF


      IF (IZMAX.EQ.0) THEN
           IZUP=KZUP
      ELSE
           IZUP=MIN(IZMAX,KZUP)
      ENDIF

C     Now check energy limits:


      DE=(EUPPER/ELOWER)**(1./(M-1.))


      IF (ELOWER.GE.EMINCUT) THEN
```

```
          KMIN=1
      ELSE
          KMIN=1+IFIX(ALOG(EMINCUT/ELOWER)/ALOG(DE))
      ENDIF


      IF (EUPPER.LE.EMAXCUT) THEN
          KMAX=M
      ELSE
          KMAX=2+IFIX(ALOG(EMAXCUT/ELOWER)/ALOG(DE))
          IF (KMAX.GT.M) KMAX=M
      ENDIF

      KMIN1=MIN(KMIN,KMAX)
      KMAX1=MAX(KMIN,KMAX)

      DO 200 J=IZLO,IZUP
         DO 150 K=KMIN1,KMAX1
            FLUX(J,K)=FLUXDUM(J,K)
150      CONTINUE
200   CONTINUE

      RETURN
      END
```

```
      SUBROUTINE UNLOAD_PATH(IPATH,UPATH,TARGET,PATH,PSTEPMIN,PSTEPMAX,
     &                        PSTEP)
C
      IMPLICIT NONE
      INTEGER*4 IPATH,NSTEP
      CHARACTER*12 TARGET
      REAL*4 UPATH,PATH,PSTEP,PSTEPMIN,PSTEPMAX,ALDEN
      DATA ALDEN/2.702000/
C
C     NOTE: current version of UNLOAD_PATH supports Al shielding only.
C

      IF (TARGET.NE.'ALUMINUM' .and. TARGET.NE.'aluminum') THEN
          WRITE(6,9999)
 9999     FORMAT(1X,' Specified shielding material unknown.  STOP')
          STOP
      ENDIF

C     Convert input UPATH (which can be in mils, cm, or g/cm2 Al)
C     to g/cm2 Al.
C
      IF (IPATH.LT.0 .or. IPATH.GT.2) THEN
          WRITE(6,9998) IPATH
 9998       FORMAT('@ 04001 ABNORMAL TERMINATION: ',
     &      /,1x,' ERROR in UNLOAD_PATH: ',
     &      /,1x,' PATH UNITS STEERING CODE UNKNOWN: ',I5,
     &      /,1x,' STOP.')
          STOP
      ENDIF


      IF (IPATH.EQ.0) THEN
C         Already specified in g/cm2
          PATH=UPATH
      ELSEIF (IPATH.EQ.2) THEN
C         Specified in cm:
          PATH=UPATH*ALDEN
      ELSEIF (IPATH.EQ.1) THEN
C         Specified in mils:
          PATH=ALDEN*2.54*UPATH/1000.
      ENDIF
C
C     Now set PSTEP for transport.

         PSTEP=PSTEPMIN

 100     CONTINUE
         NSTEP=PATH/PSTEP
         IF (NSTEP.GT.20) THEN
             PSTEP=PSTEP+0.10
             GOTO 100
         ENDIF
         IF (PSTEP.GT.PSTEPMAX) PSTEP=PSTEPMAX

C     Allow for very short PATHs:
      IF (PATH.LT.PSTEP) PSTEP=PATH

      RETURN
      END
```

```
      SUBROUTINE UNLOAD_PROTON_SPECTRUM(PROTON_FILE,EN,FLUX,NPTS)
C
C     Subroutine to unload proton Spectrum from input file into
C     array.  Can handle either CREME96 format or the old CREME format
C     (ie., two-column table of energies (in MeV) and fluxes (in
C     protons/m2-s-sr-MeV)).  CREME96 format denoted .flx, .tfx, or
C     .trp
C
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C     Last update:   17 November 1997: add .trp option
C
C
C-----------------------------------------------------------------------
C
C
      IMPLICIT NONE
      CHARACTER*80 PROTON_FILE,ILINE
      INTEGER*4 MAXSPEC,N,NZ,NZT,MM,I,NPTS,ILONG,IZ
      INTEGER*4 IVER,J,NHEADER,STAT,CREME96_OPEN
      REAL*4 EN,FLUX,EL,EU,DUMFLUX
      PARAMETER (MAXSPEC=5000)
      DIMENSION EN(1),FLUX(1),DUMFLUX(MAXSPEC)
      CHARACTER*3 SUFFIX

      CALL CHECK_CREME96_VERSION(PROTON_FILE,IVER)

c      OPEN(UNIT=10,FILE='USER:'//PROTON_FILE,STATUS='OLD',READONLY)
      stat = creme96_open(proton_file,'user',10,'old')
      ILONG=INDEX(PROTON_FILE,'.')
      SUFFIX=PROTON_FILE(ILONG+1:ILONG+3)
      CALL CAPITALIZE_STRING(SUFFIX,3)

C     Filename check re-written AJT 12-9-97
      IF (SUFFIX      .EQ. 'TFX' .or.
     &    SUFFIX      .EQ. 'FLX' .or.
     &    SUFFIX(1:2) .EQ. 'TR') THEN

          IF (IVER.GE.102) THEN
          READ(10,*) NHEADER
          DO J=1,NHEADER
             READ(10,110) ILINE
 110         FORMAT(A80)
          ENDDO
          ENDIF

          read(10,*) el,eu,n,nz,nzt
          IF (NZ.NE.1) THEN
          WRITE(6,999) PROTON_FILE,NZ,NZT
 999      FORMAT('@ 09001 ABNORMAL TERMINATION: ',
     &           /,1x,'  ERROR IN UNLOAD_PROTON_SPECTRUM:',
     &           /,1x,'  Specified file: ',
     &           /,1x,A80,
     &           /,1x,'  includes ',I5,' .le. Z .le. ',I5,
     &           /,1x,'  and contains no protons.  STOP.')
```

```
            STOP
            ENDIF
C
C           Calculate abscissae (energy values)

            EN(1)=el
            EN(N)=eu
            do 100 i=2,N-1
            EN(i)=el*(eu/el)**(float(i-1)/float(n-1))
100         continue
C

            DO 150 IZ=NZ,NZT

C           Read blank line
            read(10,110) ILINE

c           read in the flux

            read(10,*)(dumflux(i),i=1,n)

            IF (IZ.EQ.1) THEN
                DO 140 I=1,N
                       FLUX(I)=DUMFLUX(I)
140             CONTINUE
            ENDIF

150         CONTINUE

            CLOSE(10)


        ELSE

C           Standard old-CREME two-column format
            N=1
 10         CONTINUE
            READ(10,*,END=15) EN(N),FLUX(N)
            N=N+1
            GOTO 10
 15         CONTINUE
            N=N-1
            CLOSE(10)

        ENDIF

C       Eliminate end-of-file zeroes from returned spectrum:
        NPTS=0
        DO 1000 I=1,N
           IF (FLUX(I).GT.0.0) NPTS=I
 1000   CONTINUE

        RETURN
        end
```

```
      SUBROUTINE UNLOAD_SHIELDFILE(SHIELDFILE,
     &                              IUNITS,NSHIELD,UPATH,FRACSHLD)

      IMPLICIT NONE
      CHARACTER*80 SHIELDFILE
      INTEGER*4 MAXSHIELD,IUNITS,NSHIELD,K,STAT,CREME96_OPEN
      REAL*4 UPATH,FRACSHLD,TOTAL
      PARAMETER (MAXSHIELD=500)
      DIMENSION UPATH(MAXSHIELD),FRACSHLD(MAXSHIELD)
      INTEGER*4 ISHDUNIT/15/
      INTEGER*4 IVER,NHEADER
      CHARACTER*80 ILINE
C
      CALL CHECK_CREME96_VERSION(SHIELDFILE,IVER)

c      OPEN(UNIT=ISHDUNIT,READONLY,SHARED,STATUS='OLD',
c     &     FILE='USER:'//SHIELDFILE)
      stat = creme96_open(shieldfile,'user',ishdunit,'old')
C
      IF (IVER.GT.0) THEN
            READ(ISHDUNIT,*) NHEADER
            DO K=1,NHEADER
               READ(ISHDUNIT,5) ILINE
    5          FORMAT(A80)
            ENDDO
            READ(ISHDUNIT,*) IUNITS
      ENDIF

      K=0
      TOTAL=0
   10 CONTINUE
      K=K+1
      IF (K.GT.MAXSHIELD) GOTO 100
      READ(ISHDUNIT,*,END=100) UPATH(K),FRACSHLD(K)
      TOTAL=TOTAL+FRACSHLD(K)
      GOTO 10
  100 CONTINUE
      NSHIELD=K-1

      WRITE(6,999) NSHIELD
  999 FORMAT(' No. Shielding Bins  = ',I4)

C     Renormalize shielding fraction to unit integral:
      DO 200 K=1,NSHIELD
             FRACSHLD(K)=FRACSHLD(K)/TOTAL
  200 CONTINUE

      CLOSE(ISHDUNIT)

      RETURN
      END
```

```
      SUBROUTINE UNLOAD_STABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,SP)
      IMPLICIT NONE
      REAL*4 ELOWER,EUPPER
      INTEGER*4 M,IZLO,IZUP,NELM,MARR,STAT,CREME96_OPEN
      CHARACTER*12 TARGET
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 SP(NELM,MARR)
      CHARACTER*12 TARGET$
      REAL*4 ELOWER$,EUPPER$
      INTEGER*4 M$,IZLO$,IZUP$
      INTEGER*4 J,K
      CHARACTER*80 STABLEFILE

C     First, check standard table:

c      STABLEFILE='CREME96:STABLE.STD'
c      IF (IZLO.GT.28 .or. IZUP.GT.28) STABLEFILE='CREME96:STABLE.XTD'
      STABLEFILE='STABLE.STD'
      IF (IZLO.GT.28 .or. IZUP.GT.28) STABLEFILE='STABLE.XTD'

c      OPEN(UNIT=28,FILE=STABLEFILE,STATUS='OLD',READONLY,SHARED)
      stat = creme96_open(stablefile,'cr96tables',28,'old')
      READ(28,100) ELOWER$,EUPPER$,M$,IZLO$,IZUP$,TARGET$

      IF (ELOWER.EQ.ELOWER$ .AND. EUPPER.EQ.EUPPER$ .AND. M.EQ.M$ .AND.
     &    TARGET.EQ.TARGET$ .AND.
     &   (IZLO$.LE.IZLO .AND. IZLO.LE.IZUP$) .AND.
     &   (IZLO$.LE.IZUP .AND. IZUP.LE.IZUP$)) THEN
C
C     Standard table contains the necessary information.

C      WRITE(6,999) STABLEFILE(1:20), TARGET
 999  FORMAT(1x,' Standard table ',A20,' of stopping power',
     &       ' in ',A12,' used',/,1x,' for LET calculation.')

      GOTO 500
      ENDIF

C
C     Check if appropriate table exists in user area:

      CLOSE(28)
c      OPEN(UNIT=28,FILE='USER:STABLE.DAT',STATUS='OLD',READONLY,ERR=50)
      stat = creme96_open('stable.dat','user',28,'old')
      if (stat .ne. 0) goto 50
      READ(28,100) ELOWER$,EUPPER$,M$,IZLO$,IZUP$,TARGET$

      IF (ELOWER.EQ.ELOWER$ .AND. EUPPER.EQ.EUPPER$ .AND. M.EQ.M$ .AND.
     &    TARGET.EQ.TARGET$ .AND.
     &   (IZLO$.LE.IZLO .AND. IZLO.LE.IZUP$) .AND.
     &   (IZLO$.LE.IZUP .AND. IZUP.LE.IZUP$)) THEN
C
C     USER table contains the necessary information.

      WRITE(6,998) TARGET
 998  FORMAT(' User table (USER:STABLE.DAT) of stopping power'
     &     /,' in ',A12,' used for LET calculation.')

      GOTO 500
```

```
      ELSE
   50 CONTINUE
C
C     Create new STABLE.DAT in the users area:

      CLOSE(28)
      WRITE(6,997)
  997 FORMAT(' Non-standard energy or material in LETSPEC calculation.',
     &     /,' Create new stopping power table in USER area.')
      CALL STABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET)
c      OPEN(UNIT=28,STATUS='OLD',FILE='USER:STABLE.DAT',READONLY)
      stat = creme96_open('stable.dat','user',28,'old')
      READ(28,100) ELOWER$,EUPPER$,M$,IZLO$,IZUP$,TARGET$

      ENDIF

  500 CONTINUE

      READ(28,100)
      DO J=IZLO$,IZUP$
         READ(28,*) (SP(J,K),K=1,M$)
         READ(28,*)
      ENDDO
      CLOSE(28)

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,2X,1PE10.4)
200   FORMAT((1X,6(1PE10.4,2X)))

      RETURN
      END
```

```
      SUBROUTINE UNLOAD_XSECT_FILE(XSECT_FILE,NSV,XV,YV)
C
C     Subroutine to unload cross-section table from input file into
C     array.  Table is assumed to be in a two-column format, ordered
C     according to increasing first-column value.
C
C
C     Written by:    Allan J. Tylka
C                    Code 7654
C                    Naval Research Laboratory
C                    Washington, DC 20375-5352
C                    tylka@crs2.nrl.navy.mil
C
C     Last update:   14 May 1996:
C                    add 'USER:' to input file name.
C
C-----------------------------------------------------------------------------
C
C
      IMPLICIT NONE
      CHARACTER*80 XSECT_FILE
      INTEGER*4 NSV,I,STAT,CREME96_OPEN
      REAL*4 XV,YV
      DIMENSION XV(1),YV(1)

c      OPEN(UNIT=10,FILE='USER:'//XSECT_FILE,STATUS='OLD',READONLY)
      stat = creme96_open(xsect_file,'user',10,'old')
      I=1
 10   CONTINUE
      READ(10,*,END=15) XV(I),YV(I)
      I=I+1
      GOTO 10
 15   CONTINUE
      NSV=I-1
      CLOSE(10)

      RETURN
      end
```

```
      SUBROUTINE UNLOAD_ZTABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,PSTEP,
     &                              ZZ)
C
C     Subroutine used by UPROPI to unload stopping power tables into array ZZ
C
      IMPLICIT NONE
      INTEGER*4 MARR,NELM,STAT,CREME96_OPEN
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 ZZ(MARR,2,NELM)
      REAL*4 ELOWER,EUPPER,PSTEP,ELOWER$,EUPPER$,PSTEP$
      REAL*4 DELTA_PSTEP
      INTEGER*4 M,IZLO,IZUP,M$,IZLO$,IZUP$,MM,J,K
      CHARACTER*12 TARGET,TARGET$
      CHARACTER*80 ZTABLEFILE

C     FORMAT Statements

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,2X,1PE10.4)
200   FORMAT((1X,6(1PE10.4,2X)))

C     First, check standard table:

c      ZTABLEFILE='CREME96:ZTABLE.STD'
c      IF (IZLO.GT.28 .or. IZUP.GT.28) ZTABLEFILE='CREME96:ZTABLE.XTD'
      ZTABLEFILE='ZTABLE.STD'
      IF (IZLO.GT.28 .or. IZUP.GT.28) ZTABLEFILE='ZTABLE.XTD'

c      OPEN(UNIT=35,FILE=ZTABLEFILE,STATUS='OLD',READONLY,SHARED)
      stat = creme96_open(ztablefile,'cr96tables',35,'old')
      READ(35,100) ELOWER$,EUPPER$,M$,IZLO$,IZUP$,TARGET$,PSTEP$
      DELTA_PSTEP=ABS(PSTEP-PSTEP$)
C      TYPE *,' PSTEP, PSTEP$: ',PSTEP,PSTEP$
      IF (ELOWER.EQ.ELOWER$ .AND. EUPPER.EQ.EUPPER$ .AND. M.EQ.M$ .AND.
     &    TARGET.EQ.TARGET$ .AND. DELTA_PSTEP.LE.0.001 .AND.
     &   (IZLO$.LE.IZLO .AND. IZLO.LE.IZUP$) .AND.
     &   (IZLO$.LE.IZUP .AND. IZUP.LE.IZUP$)) THEN
C
C     Standard table contains the necessary information.
C
C      WRITE(6,999) ZTABLEFILE(1:20),TARGET
 999  FORMAT(' Standard table ',A20,' of stopping power',
     &       ' in ',A12,' used for',/,' transport calculation.')

      GOTO 500
      ENDIF
C
C     Check if appropriate table exists in USER area:
      CLOSE(35)
c      OPEN(UNIT=35,FILE='USER:ZTABLE.DAT',STATUS='OLD',READONLY,ERR=50)
      stat = creme96_open('ztable.dat','user',35,'old')
      if (stat .ne. 0) goto 50
      READ(35,100) ELOWER$,EUPPER$,M$,IZLO$,IZUP$,TARGET$,PSTEP$

      DELTA_PSTEP=ABS(PSTEP-PSTEP$)

      IF (ELOWER.EQ.ELOWER$ .AND. EUPPER.EQ.EUPPER$ .AND. M.EQ.M$ .AND.
     &    TARGET.EQ.TARGET$ .AND. DELTA_PSTEP.LE.0.001 .AND.
     &   (IZLO$.LE.IZLO .AND. IZLO.LE.IZUP$) .AND.
     &   (IZLO$.LE.IZUP .AND. IZUP.LE.IZUP$)) THEN
C
```

```
C     User table contains the necessary information.

      WRITE(6,998) TARGET
  998 FORMAT(' User table (USER:ZTABLE.DAT) of stopping power'
     &     /,' in ',A12,' used for transport calculation.')

      GOTO 500


      ELSE
 50   CONTINUE
C
C     Create new ZTABLE.DAT in the users area:

      CLOSE(35)
      WRITE(6,997)
  997 FORMAT(' Non-standard energy or shielding or PSTEP'
     &        ' in transport calculation.',
     &     /,' Create new stopping power table in USER area.')
      CALL ZTABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,PSTEP)
c      OPEN(UNIT=35,STATUS='OLD',FILE='USER:ZTABLE.DAT',READONLY)
      stat = creme96_open('ztable.dat','user',35,'old')
      READ(35,100) ELOWER$,EUPPER$,M$,IZLO$,IZUP$,TARGET$,
     &        PSTEP$

      ENDIF

  500 CONTINUE
      READ(35,100)
      DO J=IZLO$,IZUP$
          READ(35,*) (ZZ(K,1,J),K=1,M$)
          READ(35,100)
          READ(35,*) (ZZ(K,2,J),K=1,M$)
          READ(35,100)
      END DO
      CLOSE(UNIT=35)

      RETURN
      END
```

```
      SUBROUTINE UPROP96(INPUT_FLUX,
     &                   ELOWER,EUPPER,M,IZLO,IZUP,
     &                   NDUM,NSP,PATH,PSTEP,TARGET,
     &                   OUTPUT_FLUX)
C
C**************************************************************************
C
C This module evaluates nuclear transport, by calculating a numerical solution
C to a one dimensional continuity equation, taking into account both
C ionization energy loss (in the continuous-slowing-down approximation) and
C nuclear fragmentation.
C
C This code is based on UPROP, originally written by John R. Letaw of Severn
C Communications Corporation, working under contract to the Gamma Ray and
C Cosmic Ray Astrophysics Branch of Naval Research Laboratory in 1989.
C
C See "UPROP: A Heavy-Ion Propagation Code", by J.R. Letaw, SCC Report 89-02,
C 31 August 1989.
C
C Adapted for use with CREME96 by AJT.        Last Update  05 June 1996
C
C Important Parameters
C
C MARR    Maximum number of logarithically-spaced energy bins in spectrum
C NELM    Maximum atomic number of elements to be transported (<= 109)
C NDUM    Flag determining treatment of nuclear spallation reactions
C             If  NDUM=0 does not include nuclear spallation
C                 NDUM<0 does not follow nuclear fragments
C                 |NDUM|=1 uses energy-independent cross sections
C                 |NDUM|>1 uses cross sections calculated at N equally-spaced
C                          energies and interpolated between
C PATH    Total propagation pathlength in g/cm**2
C PSTEP   A small pathlength over which 2 nuclear fragmentations are
C         unlikely, typically 0.1 g/cm**2.
C TARGET  Name of the target shielding material
C INFILE  File containing the heavy-ion energy spectrum (<= 40 bytes)
C OUTFILE File containing the energy spectrum after transport (<= 40 bytes)
C
C Important variables
C
C FLUX    working array contains the energy spectra of up to NELM elements
C         specified at up to MARR energies
C
C**************************************************************************
      IMPLICIT NONE
      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 INPUT_FLUX(NELM,MARR),OUTPUT_FLUX(NELM,MARR)
      REAL*4 FLUX(NELM,MARR)
      REAL*4 ELOWER,EUPPER,PATH,PSTEP
      INTEGER*4 M,N,NDUM,IZLO,IZUP,J,K,NSP
      CHARACTER TARGET*12
C
C     Copy input fluxes to working array:
      DO 20 J=IZLO,IZUP
         DO 10 K=1,M
            FLUX(J,K)=INPUT_FLUX(J,K)
 10      CONTINUE
 20   CONTINUE
C
```

```
C     Special case for very thin shield (0.20 g/cm2 = 29.16 mils Al):
C     Added by AJT, per JHA suggestion, 6-5-96
C
      IF (PATH.LT.0.20) THEN
          CALL THIN_SHIELD(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,PATH,FLUX)
      ELSE

      N=NDUM
      IF (PATH.LT.0.1) N=0

      IF (N.NE.0) THEN

C     Alternate ionization loss and fragmentation using the pathlength
C     PSTEP until PATH is accumulated.

        DO J=1,INT(PATH/PSTEP+0.5)
          CALL UPROPI(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,PSTEP,FLUX)
          CALL UPROPC(ELOWER,EUPPER,M,N,NSP,IZLO,IZUP,TARGET,PSTEP,FLUX)
        END DO

      ELSE

C     Do ionization loss (only) using the pathlength PSTEP until PATH
C     is accumulated

        DO J=1,INT(PATH/PSTEP+0.5)
          CALL UPROPI(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,PSTEP,FLUX)
        END DO

      ENDIF

      ENDIF

C     Copy transported energy spectra to output arrays:

      DO 200 J=IZLO,IZUP
         DO 100 K=1,M
            OUTPUT_FLUX(J,K)=FLUX(J,K)
  100    CONTINUE
  200 CONTINUE

      RETURN
      END
```

```fortran
      SUBROUTINE UPROPC(ELOWER,EUPPER,M,N,NSP,IZLO,IZUP,TARGET,
     &      PSTEP,FLUX)
C***********************************************************************
C SUBROUTINE UPROPC in Module UPROP.FOR
C
C Nuclear spallation subroutine.  Determines the attenuation of a heavy-ion
C energy spectrum from spallation reactions in passage through shielding
C material.  Initiates creation of an auxiliary data file (CTABLE.DAT) if
C a suitable one does not already exist.
C Modified by A.F. Barghouty 3-25-96
C***********************************************************************
C Parameters
C
C MARR    Maximum number of logarithically-spaced energy bins in spectrum
C NELM    Maximum atomic number of elements to be transported (<= 109)
C MCS     Maximum number of energies at which cross section data are
C         defined
C ELOWER  Lower energy bound of input and output spectra (>= 0.1 MeV)
C EUPPER  Upper energy bound of input and output spectra (<= 100000 MeV)
C M       Number of logarithmically equally-spaced energy bins (<= MARR)
C N       Number of energies at which cross section data are defined (<= MCS)
C IZLO    Least atomic number of elements transported (>= 1)
C IZUP    Greatest atomic number of elements transported (<= 109)
C TARGET  Name of the target shielding material (<= 12 bytes)
C PSTEP   A small pathlength over which 2 nuclear fragmentations are
C         unlikely, typically 0.1 g/cm**2.
C FLUX    Contains the energy spectra of elements IZLO through IZUP at
C         M energies
C
C Important  variables
C
C E       Energy at each M-point grid (spectrum grid)
C ECS     Energy at each N-point grid (cross section grid)
C VC      Version number of current spallation cross section data file
C         (CTABLE.DAT)
C FT      Temporary flux vector used in calculating secondary spectra
C CC      Partial and total cross section data computed by SPROP and stored
C         in CTABLE.DAT.  First index is product Z; second index is target
C         Z; third index is index in N-point energy grid.
C CCT     Temporary array for holding cross sections at current M-point
C         energy grid.
C REL     Factor for normalizing energy to minimum energy on the grid
C FUL     Factor relating energy ratio to number of bins on the grid
C***********************************************************************
      PARAMETER (MARR=5000,NELM=92,MCS=10)

      REAL*4 FLUX(NELM,MARR),CC(NELM,NELM,MCS),E(MARR),ECS(MCS)
      REAL*4 FT(NELM),CCT(NELM,NELM),SUM1(NELM),SUM2(NELM)
      REAL*4 SPLOSS(NELM,MCS),SPLT(NELM,MARR),dFLUX(NELM,MARR)
      REAL*4 FACTOR(NELM),TFLUX1(NELM),TFLUX0(NELM)
      CHARACTER*12 TARGET,TARGET$
      DATA IENT/0/

      IF (IENT.EQ.0) THEN
          IENT=1
          CALL UNLOAD_CTABLE(ELOWER,EUPPER,N,NSP,IZLO,IZUP,TARGET,
     &                          CC,SPLOSS)
      ENDIF

      NABS=ABS(N)
```

```
C     Compute energies associated with flux points

      REL=1./ELOWER
      FUL=1./LOG(EUPPER/ELOWER)
      DE=(EUPPER/ELOWER)**(1./(FLOAT(M)-1.))
      E(1)=ELOWER
      DO I=2,M-1
        E(I)=E(I-1)*DE
      END DO
      E(M)=EUPPER

C     Compute energies associated with cross sections

      IF (NABS.GE.2) THEN
        DE=(EUPPER/ELOWER)**(1./(FLOAT(NABS)-1.))
        ECS(1)=ELOWER
        DO I=2,NABS-1
          ECS(I)=ECS(I-1)*DE
        END DO
        ECS(NABS)=EUPPER
      END IF
C
C     Perform fragmentation for each particle energy
C
      DO I=1,M

C       Perform linear interpolation of cross
C       section matrix appropriate for current energy
C
C...     [Interpolation debugged -for large II- 3/23/1996]
C
        IF (NABS.GE.2) THEN
          XI=1.+(NABS-1.)*LOG(E(I)*REL)*FUL
          II=INT(XI)
          IF (XI.GE.NABS) THEN
           DELTA=1./(ECS(NABS)-ECS(NABS-1))
          ELSE
            DELTA=1./(ECS(II+1)-ECS(II))
          ENDIF
          IF(I.LT.M) THEN
           F1X=DELTA*(E(I)-ECS(II))
           F2X=DELTA*(ECS(II+1)-E(I))
          END IF
          DO J=IZLO,IZUP
            DO K=IZLO,IZUP
              IF(I.EQ.1) THEN
                CCT(J,K)=CC(J,K,1)
                SPLT(J,I)=SPLOSS(J,1)
              END IF
              IF(I.LT.M) THEN
                CCT(J,K)=F1X*CC(J,K,II+1)+F2X*CC(J,K,II)
                SPLT(J,I)=F1X*SPLOSS(J,II+1)+F2X*SPLOSS(J,II)
              END IF
              IF(I.EQ.M) THEN
                CCT(J,K)=CC(J,K,NABS)
                SPLT(J,I)=SPLOSS(J,NABS)
              END IF
            END DO
          END DO
```

```
        ELSE
          DO J=IZLO,IZUP
            DO K=IZLO,IZUP
              CCT(J,K)=CC(J,K,1)
              SPLT(J,I)=SPLOSS(J,1)
            END DO
          END DO
        ENDIF
C
        IF (N.GT.0) THEN

C         If N > 0, compute fragmentation losses and gains

C         Form a temporary flux vector and multiply by PSTEP
C         Secondaries (only) are computed from new vector

          DO J=IZLO,IZUP
            FT(J)=FLUX(J,I)*PSTEP
          END DO
C
C         Modify flux according to secondary production (includes
C         all losses and gains)

          DO J=IZLO,IZUP
            DO K=IZLO,IZUP
              FLUX(J,I)=FLUX(J,I)+CCT(J,K)*FT(K)
            END DO
            IF (FLUX(J,I).LT.1.E-20) FLUX(J,I)=0.
          END DO
C
        ELSE IF (N.LT.0) THEN

C         If N < 0, compute only fragmentation loss

          DO J=IZLO,IZUP
            FLUX(J,I)=FLUX(J,I)*(1.+CCT(J,J)*PSTEP)
            IF (FLUX(J,I).LT.1.E-20) FLUX(J,I)=0.
          END DO

        ENDIF
      END DO
C
C         Compute new flux taking into account energy losses due to
C         fragmentation:                (Sept. 1993)
C
          IF(NSP.EQ.1) THEN
           IF(IENT.EQ.0) THEN
           IENT=1
           WRITE(6,9999)
 9999      FORMAT(1x,' UPROPC: Straight-ahead approx. NOT used.')
           END IF
C
          DO I=IZLO,IZUP
             SUM1(I)=0.
             DO J=1,M
               SUM1(I)=SUM1(I)+FLUX(I,J)*E(J)
             END DO
          END DO
C
          DO I=IZLO,IZUP
```

```
            DO J=1,M
              dFLUX(I,J)=0.
C
              IF(J.EQ.1) THEN
              dEN=1./(E(2)-E(1))
              dFLUX(I,1)=dEN*(SPLT(I,1)*FLUX(I,2)+SPLT(I,2)*FLUX(I,1)
     &                   -2.*FLUX(I,1)*SPLT(I,1))
C
              ELSE IF(J.EQ.M) THEN
              dEN=1./(E(M)-E(M-1))
              dFLUX(I,M)=dEN*(2.*FLUX(I,M)*SPLT(I,M)
     &                   -SPLT(I,M-1)*FLUX(I,M)-SPLT(I,M)*FLUX(I,M-1))
C
              ELSE
              dEN=1./(E(J+1)-E(J-1))
              dFLUX(I,J)=dEN*((SPLT(I,J+1)-SPLT(I,J-1))*FLUX(I,J)
     &                      +(FLUX(I,J+1)-FLUX(I,J-1))*SPLT(I,J))
C
              END IF
C
              IF(ABS(dFLUX(I,J)).LT.1.E-20) dFLUX(I,J)=0.
              FLUX(I,J)=FLUX(I,J)+dFLUX(I,J)*PSTEP
              IF(FLUX(I,J).LT.1.E-20) FLUX(I,J)=0.
C
              END DO
            END DO
C
           DO I=IZLO,IZUP
              SUM2(I)=0.
              DO J=1,M
               SUM2(I)=SUM2(I)+FLUX(I,J)*E(J)
              END DO
           END DO
C
C.........Normalization:
           DO I=IZLO,IZUP
              ZERO=ABS(1.-SUM1(I)/SUM2(I))
              IF(ZERO.GT..01) THEN
C                TYPE *,' '
C                TYPE *,'    *** Normalization...! ***'
C                TYPE *,' '
              END IF
           END DO
C
           END IF
        RETURN
        END
```

```
      SUBROUTINE UPROPI(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,
     &       PSTEP,FLUX)

C***********************************************************************
C SUBROUTINE UPROPI in Module UPROP.FOR
C
C Ionization loss subroutine.  Determines the attenuation of a heavy-ion
C energy spectrum from ionization losses in passage through shielding
C material.  Initiates creation of an auxiliary data file (ZTABLE.DAT) if
C a suitable one does not already exist.
C***********************************************************************
C Parameters
C
C MARR    Maximum number of logarithically-spaced energy bins in spectrum
C NELM    Maximum atomic number of elements to be transported
C ELOWER  Lower energy bound of input and output spectra (<= 0.1 MeV)
C EUPPER  Upper energy bound of input and output spectra (>= 100000 MeV)
C M       Number of logarithmically equally-spaced energy bins (<= MARR)
C IZLO    Least atomic number of elements transported (>= 1)
C IZUP    Greatest atomic number of elements transported (<= 109)
C TARGET  Name of the target shielding material (<= 12 bytes)
C PSTEP   A small pathlength over which 2 nuclear fragmentations are
C         unlikely, typically 0.1 g/cm**2.
C FLUX    Contains the energy spectra of elements IZLO through IZUP at
C         M energies
C
C Important  variables
C
C FLUX2   Temporary vector containing the energy spectrum of a single
C         element
C ZZ      Range and stopping power data computed by ZPROP and stored in
C         ZTABLE.DAT
C REL     Factor for normalizing energy to minimum energy on the grid
C FUL     Factor relating energy ratio to number of bins on the grid
C***********************************************************************

      IMPLICIT NONE
      INTEGER*4 MARR,NELM
      PARAMETER (MARR=5000,NELM=92)
      REAL*4 FLUX(NELM,MARR),ZZ(MARR,2,NELM),E(MARR),FLUX2(MARR)
      CHARACTER*12 TARGET
      INTEGER*4 IENT,M,IZLO,IZUP,I,J,K,KK
      REAL*4 ELOWER,EUPPER,PSTEP,REL,FUL,DE,XK
      DATA IENT/0/

      IF (IENT.EQ.0) THEN
          IENT=1
          CALL UNLOAD_ZTABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,PSTEP,ZZ)
      ENDIF

C     Compute new flux

      REL=1./ELOWER
      FUL=1./LOG(EUPPER/ELOWER)
      DE=(EUPPER/ELOWER)**(1./(M-1.))
      E(1)=ELOWER
      DO I=2,M-1
        E(I)=E(I-1)*DE
      END DO
      E(M)=EUPPER
```

```
      DO J=IZLO,IZUP
        DO K=1,M
          XK=1.+(M-1.)*LOG(ZZ(K,1,J)*REL)*FUL
          KK=INT(XK)
          IF (XK.GE.M) THEN
            FLUX2(K)=((ZZ(K,1,J)-E(M-1))*FLUX(J,M)+
     &                (E(M)-ZZ(K,1,J))*FLUX(J,M-1))/(E(M)-E(M-1))
          ELSE
            FLUX2(K)=((ZZ(K,1,J)-E(KK))*FLUX(J,KK+1)+
     &                (E(KK+1)-ZZ(K,1,J))*FLUX(J,KK))/(E(KK+1)-E(KK))
          ENDIF
          FLUX2(K)=FLUX2(K)*ZZ(K,2,J)
          IF (FLUX2(K).LT.1.E-20) FLUX2(K)=0.
        END DO
        DO K=1,M
          FLUX(J,K)=FLUX2(K)
        END DO
      END DO
      RETURN
      END
```

```
CC*********************************************************************
c
c       Module: VAX_ROUTINES
c
c
c       Logical Names and Environment Variables serve the same purpose,
c       but are handled differently, on the two CREME96 platforms (VAX and
c       PC respectively).  There are also differences between the two file
c       OPEN statements.  To enable platform independance where fully
c       specified filenames and where file opens are used in the higher
c       level CREME96 code, two versions exist of the routines to perform
c       these tasks.  When an executable is being created, it is the
c       responsibility of the person performing the link to ensure that the
c       appropriate set of routines is used for the current build.
c
c
c       Three platform-DEPENDANT routines exist:
c
c               CREME96_FULL_FILENAME   creates fully-specified filename
c               CREME96_OPEN            performs a file OPEN on full filename
c
c               CHECK_FILE.FOR          Added 6/8/97
c                                       because VAX version contains LIB$SPAWN
C                                       REMOVED 11/18/97 by AJT
C
C               SHOW_DIRECTORY.FOR      Added 11/18/97; contains LIB$SPAWN
C
c
c       These routines reside in the following 2 physical files:
c
c               VAX_ROUTINES.FOR        used for a VAX build (this file)
c               PC_ROUTINES.FOR         used for a PC build
c
c*********************************************************************


        Integer function creme96_open(filename,path,unit,status)


c
c       FILENAME:       The non-fully specified name of the target file.
c
c       PATH:           Contains the VMS logical pointing to directory
c                       where file does, or will exist.
c
c       UNIT:           The logical unit to be associated with the file.
c                       Must be defined at the time of the function call
c                       (one will not be assinged by this routine).
c
c       STATUS:         Contains either OLD, for existing file, or
c                       NEW, to create a file.
c
c       Calling example:
c
c               STAT = creme96_open('input.dat','creme96',inunit,'old')
c
c       Success is indicated by a ZERO return value.  Otherwise, the
c       return value will contain the FORTRAN error code.
c
        IMPLICIT NONE
```

```
        character*80     file,creme96_full_filename,line
        character*(*)    filename
        character*(*)    path
        integer          unit,ios
        character*(*)    status

c       WRITE(*,*) 'IN OPEN... FILENAME: ',FILENAME,'     PATH: ',PATH

        file = creme96_full_filename(filename,path)

        if (status(1:1) .eq. 'o' .or. status(1:1) .eq. 'O') then
c          Old files are only opened for READ (no APPEND in CREME96).
c          Any file opened for READ will be opened SHARED.

           open(unit=unit,file=file,status='old',
     &     READONLY,SHARED,iostat=ios,err=199)

c DEBUG
c          read(unit,99)line
c99        format(a80)
c          write(*,*)'First line in file: ',line

        else

c          New file to be created.  WRITE and NOSHARE are default.  On the PC,
c          we must open with REPLACE instead of NEW, in case a file already
c          exists of this name (as it is our intention to write over it).  If
c          one doesn't exist, REPLACE acts the same as NEW.

           OPEN(UNIT=unit,file=file,status='new',
     &     iostat=ios,err=199)

c DEBUG
c          write(*,*)'Writing test line to new file...'
c          write(unit,*)'Test line'

        endif

199     creme96_open = ios
        return
        end
c
c**********************************************************************
c
        Character*80 function creme96_full_filename(filename,path)


c       The variable PATH contains the name of the VMS logical, which
c       in turn points to the directory path of the target file.
c       This routine appends the logical name to the bare filename.

        IMPLICIT NONE
        character*(*)    filename
        character*(*)    path


c       WRITE(*,*) 'IN FULL... FILENAME: ',FILENAME,'     PATH: ',PATH

        creme96_full_filename = path//':'//filename
```

```
c       WRITE(*,*) 'FULL_FILENAME: ',CREME96_FULL_FILENAME

        return
        end
c
c*******************************************************************
       SUBROUTINE SHOW_DIRECTORY(JFILETYPE)
C
C      VAX-specific routine.
C
       INTEGER*4 JFILETYPE,ISTAT
       LOGICAL LIBSPAWN

       IF (JFILETYPE.EQ.0) THEN
        WRITE(6,9010)
 9010   FORMAT(1x,' Here is the directory of your current USER area:')
        ISTAT=LIB$SPAWN('DIR USER:*.*')

       ELSEIF (JFILETYPE.EQ.1) THEN
        WRITE(6,9011)
 9011   FORMAT(1x,' Here are the .TRP files in your current USER area:')
        ISTAT=LIB$SPAWN('DIR USER:*.TR*')

       ELSEIF (JFILETYPE.EQ.2) THEN
        WRITE(6,9012)
 9012   FORMAT(1x,' Here are the .GTF files in your current USER area:')
        ISTAT=LIB$SPAWN('DIR USER:*.GT*')

       ELSEIF (JFILETYPE.EQ.3) THEN
        WRITE(6,9013)
 9013   FORMAT(1x,' Here are the .FLX files in your current USER area:')
        ISTAT=LIB$SPAWN('DIR USER:*.FLX')
        WRITE(6,9014)
        ISTAT=LIB$SPAWN('DIR USER:*.TFX')
        WRITE(6,9011)
        ISTAT=LIB$SPAWN('DIR USER:*.TR*')

       ELSEIF (JFILETYPE.EQ.4) THEN
        WRITE(6,9014)
 9014   FORMAT(1x,' Here are the .TFX files in your current USER area:')
        ISTAT=LIB$SPAWN('DIR USER:*.TFX')
        WRITE(6,9013)
        ISTAT=LIB$SPAWN('DIR USER:*.FLX')
        WRITE(6,9011)
        ISTAT=LIB$SPAWN('DIR USER:*.TR*')

       ELSEIF (JFILETYPE.EQ.5) THEN
        WRITE(6,9015)
 9015   FORMAT(1x,' Here are the .LET files in your current USER area:')
        ISTAT=LIB$SPAWN('DIR USER:*.LET')

       ELSEIF (JFILETYPE.EQ.6) THEN
        WRITE(6,9016)
 9016   FORMAT(1x,' Here are the .DLT files in your current USER area:')
        ISTAT=LIB$SPAWN('DIR USER:*.DLT')

       ELSEIF (JFILETYPE.EQ.7) THEN
        WRITE(6,9017)
 9017   FORMAT(1x,' Here are the .SHD files in your current USER area:')
        ISTAT=LIB$SPAWN('DIR USER:*.SHD')
```

```
      ELSEIF (JFILETYPE.EQ.8) THEN
       WRITE(6,9018)
9018   FORMAT(1x,' Here are the .XSD files in your current USER area:')
       ISTAT=LIB$SPAWN('DIR USER:*.XSD')

      ENDIF

      WRITE(6,9999)
9999  FORMAT(/)
      RETURN
      END
```

```
      REAL*4 FUNCTION WEIBULL(ONSET,WIDTH,POWER,ASYMPTOTE,E)
C
C
C      Returns value of Weibull cross-section evaluated at abscissa E
C      This function can be used for either heavy-ion or proton
C      cross-sections;  but the units are different in each case.
C
C      Input parameters of Weibull fit:
C         O= onset     (in MeV for proton; in MeV-cm2/mg for heavy ion LET)
C         W= width     (as above)
C         P= power     (dimensionless exponent)
C         A= asymptote (in 10E-12 cm2/bit for protons cross-sections;
C                       in square microns/bit for heavy-ion cross-sections)
C         E= absicissa (in MeV for protons; in MeV-cm2/mg for heavy ion LET)
C
C      Output:  SEU cross-section (same units as asymptote)
C
C      Written by:    Allan J. Tylka
C                     Code 7654
C                     Naval Research Laboratory
C                     Washington, DC 20375-5352
C                     tylka@crs2.nrl.navy.mil
C
C      Last update:   29 March 1996
C
C-----------------------------------------------------------------------
C

      IMPLICIT NONE
      REAL*4 E,ONSET,Y,WIDTH,POWER,ASYMPTOTE

      WEIBULL=0
      IF (E.LT.ONSET) RETURN
      Y = ((E-ONSET)/WIDTH)**POWER
      Y=1.0-exp(-Y)
      WEIBULL=ASYMPTOTE*Y
      IF (WEIBULL.LT.0.) WEIBULL=0.
      RETURN
      END
```

```
      SUBROUTINE YIELDX(IZ, IA, JZ, JA, EJ, QJ)

C..............................................................
C..............................................................
C...                                                        ...
C...  Silberberg&Tsao Semiempirical Cross Sections Routines.  ...
C...  Notation:                                             ...
C...  The S&T routines give the inelastic cross section QJ (in mb)  ...
C...  for the reaction;                                     ...
C...  (IZ,IA) + proton --> (JZ,JA) at energy EJ (in MeV/Nucleon),  ...
C...  on the basis of a set of semiempirical formulae.      ...
C...  Inputs:  IZ - Atomic number of incoming nucleus        ...
C...           IA - Atomic weight of incoming nucleus        ...
C...           JZ - Atomic number of secondary nucleus       ...
C...           JA - Atomic weight of secondary nucleus       ...
C...           EJ - Amount of energy per nucleon of secondary nucleus...
C...  OUTPUT:  QJ - Inelastic cross section of reaction      ...
C...  References in commented lines are from:               ...
C...  [1] Silberberg, R. & Tsao, C. H. 1973, Astrophys. J. Suppl.,  ...
C...                                   25, pp. 315 - 333.   ...
C...  [2] Silberberg, R. & Tsao, C. H. 1973, Astrophys. J. Suppl.,  ...
C...                                   25, pp. 335 - 368.   ...
C...  [3] Silberberg, R., Tsao, C. H. & Letaw, J. R.,       ...
C...                 Astrophys. J. Suppl., 58, pp. 873 - 881.  ...
C...                                                        ...
C...  Another useful reference is:                          ...
C...      Silberberg, R. & Tsao, C. H. 1990, Phys. Rep., 191, 351.  ...
C...                                                        ...
C...               *** Rewritten November 1995 ***          ...
C...  Majority of comments written by Mark E. Mattson,      ...
C...  March - April, 1996, e-mail:  mattson@vpihe4.phys.vt.edu  ...
C...                                                        ...
C...  Routines linked:                                      ...
C...  (1) PXN; for JZ = IZ+1 but JA=IA (i.e., pick-up reactions)  ...
C...  (2) YIELD1 (through 4 are spallation reactions); for JZ <= 4,  ...
C...  (3) YIELD2; for JZ >= 5 and  5 <= IZ <= 16            ...
C...  (4) YIELD3; for JZ >= 5 and 17 <= IZ <= 20            ...
C...  (5) YIELD4; for JZ >= 5 and 21 <= IZ <= 92            ...
C...  (6) CORRECTIONS; for various energy and/or structure-related  ...
C...                 correction factors to spallation yields.  ...
C...                                                        ...
C..............................................................
C..............................................................

      IMPLICIT NONE
      REAL*4 EJ,QJ
      INTEGER*4 IZ,IA,JZ,JA
      REAL*4 Q1,Q2,Q3,Q4,EXPF
      REAL*4 QR,QE,QF,QH,FE,FF,FA,FZ,CJ,PN,GA,ANZJ,AA,AE,AC,EC

      LOGICAL NULL,NOT_CONSERVED,OUT_OF_RANGE,
     &        NON_PICKUP,PICKUP,REJECTED
      LOGICAL REGION1,REGION2,REGION3

      COMMON /FS/ QR,QE,QF,QH,FE,FF,FA,FZ,CJ,PN,GA,ANZJ,AA,AE,AC,EC

      QJ = 0

C... Definitions:
C... This is in case the input indicates the secondary nucleus is exactly
```

```
C... the same as the initial nucleus.
      NULL = IZ*IA.EQ.JZ*JA

C... If the change in the atomic number of the nuclei is greater than the
C... change in atomic number, then there's something wrong with the input.
      NOT_CONSERVED = ((IZ-JZ).GT.(IA-JA)) .OR. NULL

C... If the incoming atomic weight is less than the secondary atomic weight,
C... or if the incoming atomic number is greater than 92 (uranium), of if
C... the incoming atomic number indicates hydrogen or helium, we're not
C... interested in it.
      OUT_OF_RANGE = IA.LT.JA .OR. IZ.GT.92 .OR. IZ.LE.2

C... If the secondary atomic number is 2 or more greater than the incoming
C... atomic number, or if the number of neutrons in the secondary nucleus
C... is greater than the number of neutrons in the incoming nucleus, or if
C... the secondary nucleus is that of hydrogen, we're not interested in it.
      NON_PICKUP = (IZ-JZ).LT.-1 .OR. ((IA-IZ)-(JA-JZ)).LT.0
     &                            .OR. JZ.LE.1

C... If the atomic number of the secondary is 1 greater than the atomic
C... number of the incoming and if the incoming atomic weight is at least
C... as great as the secondary atomic weight, we ARE interested in it, but
C... only as a (p,pxn) "pick-up" reaction.
      PICKUP = (IZ-JZ).EQ.-1 .AND. IA.GE.JA

C... Regions of applicability as determined by the incoming nucleus:
      REGION1 = IZ.GE. 5 .AND. IZ.LE.16 ! Nuclei for Boron through Sulphur
      REGION2 = IZ.GE.17 .AND. IZ.LE.20 ! Nuclei for Chlorine through Calcium
      REGION3 = IZ.GE.21 .AND. IZ.LE.92 ! Nuclei for Scandium through Uranium

C... Initial Rejections:
      REJECTED = NULL.
     &         OR.NOT_CONSERVED.
     &         OR.OUT_OF_RANGE.
     &         OR.NON_PICKUP
      IF (REJECTED) RETURN

C... Non-Spallation, but pick-up reactions:
      IF (PICKUP) THEN
         CALL PXN(REAL(IZ),REAL(IA),REAL((IA-IZ)-(JA-JZ)),EJ,QJ)
         RETURN
      END IF

C... Spallation reactions sorted according to Z-number of secondary:
      IF (JZ.LE.4) THEN    ! If the secondary nucleus is between H and Be
         CALL YIELD1(IZ, IA, JZ, JA, EJ, QJ)
      ELSE
         IF (REGION1) CALL YIELD2(IZ, IA, JZ, JA, EJ, QJ)
         IF (REGION2) CALL YIELD3(IZ, IA, JZ, JA, EJ, QJ)
         IF (REGION3) CALL YIELD4(IZ, IA, JZ, JA, EJ, QJ)
      END IF

C...  Apply energy and/or structure-related correction factors:
      CALL CORRECTIONS(IZ, IA, JZ, JA, EJ, QJ)

      RETURN
      END
C
C
```

```
C
C
      SUBROUTINE CORRECTIONS(IZ, IA, JZ, JA, EJ, QJ)

C.............................................................................
C.............................................................................
C...                                                                       ...
C...  This subroutine includes corrections for both the energy and        ...
C...  structure functions.  Much of this is outlined in ref [3].          ...
C...  Latest corrections, however, are outlined in Ref. 15 of CPC         ...
C...  write-up!                                                           ...
C...                                                                       ...
C.............................................................................
C.............................................................................

      IMPLICIT NONE
      REAL*4 EJ,QJ
      INTEGER*4 IZ,IA,JZ,JA
      REAL*4 Q1,Q2,Q3,Q4,EXPF
      REAL*4 QR,QE,QF,QH,FE,FF,FA,FZ,CJ,PN,GA,ANZJ,AA,AE,AC,EC
      COMMON /FS/ QR,QE,QF,QH,FE,FF,FA,FZ,CJ,PN,GA,ANZJ,AA,AE,AC,EC

C... Energy-related corrections to results of YIELD1:
      IF (JZ.LE.4) THEN
         IF (IZ.EQ.6 .AND. EJ.GT.200.0 .AND. EJ.LE.400.0)
     &      QJ = QJ*(1.0 - 0.002*(EJ - 200.0))
         IF (IZ.EQ.6 .AND. EJ.GT.400.0 .AND. EJ.LE.1000.0)
     &      QJ = QJ*0.6
         IF (IZ.EQ.6 .AND. EJ.GT.1000.0 .AND. EJ.LE.5000.0)
     &      QJ = QJ*(0.6 + 0.0001*(EJ - 1000.0))

C... Structure-related corrections to results of YIELD1:
         IF (IZ.EQ.6 .AND. IA.EQ.12 .AND. JZ.EQ.4 .AND. JA.EQ. 8)
     &      QJ = QJ*1.8
      ENDIF

C... Structure-related corrections to restuls of YIELD1, 3, and 4:
      IF (IZ.EQ.7 .AND. IA.EQ.14 .AND. JZ.EQ.6 .AND. JA.EQ.12)
     &      QJ = QJ*1.8
      IF (IZ.EQ.7 .AND. IA.EQ.14 .AND. JZ.EQ.6 .AND. JA.EQ.13)
     &      QJ = QJ*0.5
      IF (IZ.EQ.8 .AND. IA.EQ.16 .AND. JZ.EQ.6 .AND. JA.EQ.12)
     &      QJ = QJ*1.8
      IF (IZ.EQ.8 .AND. IA.EQ.16 .AND. JZ.EQ.7 .AND. JA.EQ.14)
     &      QJ = QJ*1.8
      IF (IZ.EQ.8 .AND. IA.EQ.16 .AND. JZ.EQ.7 .AND. JA.EQ.15)
     &      QJ = QJ*1.5
C
C...
C
      IF (IZ.EQ. 8 .OR.IZ.EQ.10) THEN
         IF (IZ*2.EQ.IA.AND.JA-2*JZ.GE.2.AND.JZ.GE.5)     QJ=QJ*0.7
      ENDIF
      IF (IZ.GE. 9.AND.IZ.LE.16.AND.2*JZ-JA.EQ.1.AND.JZ.GE.9)QJ=QJ*.7
      IF (IZ.GE.10.AND.IZ.LE.13.AND.IA-IZ*2.NE.2)  THEN
         IF((JZ.EQ.6.AND.JA.EQ.12).OR.(JZ.EQ.8.AND.JA.EQ.16)) QJ=QJ*2.
      ENDIF
      IF((IZ.EQ.10.AND.IA.EQ.20).or.(IZ.EQ.12.AND.IA.EQ.24)) THEN
         IF (JZ.EQ.7.AND.(JA.EQ.14.OR.JA.EQ.15))              QJ=QJ*1.5
      ENDIF
```

```
      IF (IZ.GE.10.AND.IZ.LE.16.AND.JZ.EQ.9 )             QJ=QJ*0.8
      IF ((IZ.EQ.12.OR.IZ.EQ.14.OR.IZ.EQ.16).AND.(2*IZ.EQ.IA)) THEN
         IF (IZ-JZ.EQ.2.AND.IA-JA.EQ.4)              QJ=QJ*1.6
         IF (IZ-JZ.EQ.1.AND.IA-JA.EQ.1)              QJ=QJ*1.6
      ENDIF
      IF ((IZ.EQ.18.OR.IZ.EQ.20).AND.2*IZ.EQ.IA)  THEN
        IF (IZ-JZ.EQ.1.OR.IZ-JZ.EQ.3)                QJ=QJ*0.7
      ENDIF
      IF (IZ.EQ.20.AND.IA.EQ.40.AND.(JZ.EQ.12.OR.JZ.EQ.14))QJ=QJ*2.4
      IF (IZ.EQ.20.AND.IA.EQ.40.AND.(JZ.EQ.18.OR.JZ.EQ.16))QJ=QJ*1.6
      IF (IZ.GE.24.AND.IZ.LE.28) THEN
         IF (JZ.GE.20.AND.JZ.LE.23.and.JA-JZ*2.GE.6)      QJ=QJ*0.5
      ENDIF
      IF((IZ.EQ.26.AND.IA.EQ.56).OR.(IZ.EQ.24.AND.IA.EQ.52)) THEN
         IF((JZ.EQ.20).OR.(JZ.EQ.18).OR.(JZ.EQ.16))       QJ=QJ*1.3
      ENDIF
      IF (IZ.GE.30.AND.IZ-JZ.GE.6)
     &   QJ = QJ*(1.0+0.9*EXPF(-((EJ-1230)/150)**2)
     &           *EXPF(-(ABS(IZ-JZ-12)/5.)**2))
      IF (IZ.EQ.26 .AND. IA.EQ.56 .AND. JZ.EQ.23)
     &   QJ = QJ*(1.0 - 0.6*EXPF(-((52-JA)/2.6)**2))
      IF (IZ.EQ.26 .AND. IA.EQ.56 .AND. JZ.EQ.24. AND. JA.EQ.54)
     &   QJ = 0.7*QJ
      IF (IZ.EQ.26 .AND. IA.EQ.56 .AND. JZ.EQ.25. AND. JA.GE.54 .AND.
     &   JA.LE.55) QJ = QJ*(1.7 - (JA-54)*0.45)
      IF (IZ.EQ.26 .AND. IA.EQ.56 .AND. JZ.EQ.26 .AND. JA.EQ.53)
     &   QJ = QJ*2.0
      IF (IZ.EQ.26 .AND. IA.EQ.56 .AND. JZ.EQ.17) QJ = QJ*0.9
C
      IF (JZ.EQ.5) THEN
         CALL YIELD1(IZ, IA, JZ, JA, EJ, Q1)
         QJ = SQRT(Q1*QJ)
         IF (IZ.EQ.7 .AND. IA.EQ.14 .AND. JZ.EQ.5 .AND. JA.EQ.10)
     &      QJ = QJ*1.8
         IF (IZ.EQ.6 .AND. IA.EQ.12 .AND. JZ.EQ.5 .AND. JA.EQ.10)
     &      QJ = QJ*1.8
         IF (IZ.EQ.6 .AND. IA.EQ.12 .AND. JZ.EQ.5 .AND. JA.EQ.11)
     &      QJ = QJ*1.5
      ELSE
         IF (IZ.EQ.20.AND.JZ.GE.6) THEN
            CALL YIELD4(IZ,IA,JZ,JA,EJ,Q4)
            QJ = SQRT(QJ*Q4)
         ENDIF
         IF (IZ.EQ.21 .AND. JZ.GE.6) THEN
            CALL YIELD3(IZ, IA, JZ, JA, EJ, Q3)
            QJ = SQRT(QJ*Q3)
         ENDIF
      ENDIF
      RETURN
      END
C
C
C
C
C...................................................................
C...................................................................
C...                                                             ...
C... The next four subroutines are used in the determination of the ...
C... various parameters in Table 1 of ref [1].  They are presented in...
C... this fashion to facilitate optimization with the data.       ...
```

```
C...                                                                   ...
C...         Y1BLOK - Parameters as in Table 1A of ref [1].            ...
C...         Y2BLOK - Parameters as in Table 1B of ref [1].            ...
C...         Y3BLOK - Parameters as in Table 1C of ref [1].            ...
C...         Y4BLOK - Parameters as in Table 1D of ref [1].            ...
C...                                                                   ...
C......................................................................
C......................................................................
C
C
      BLOCK DATA Y1BLOK

      COMMON /AT/ BA,C2,C3,C4,C5,QC,QL,B,PC,PL,RC,RL,RU,S,SE,C,T,ET

      DIMENSION  BA(92), C2(56), C3(56), C4(56), C5(56), ET(4)
      DATA  BA / 16 * 0.0,  36.0,  38.0,  40.0,  44.0,
     & 45.0, 48.0, 50.0, 52.0,  55.0, 55.7,   58.5,  61.0,  64.0,  67.0,
     & 70.0, 73.0, 75.0, 78.0,  80.0, 82.0,   84.0,  86.0,  89.0,  92.0,
     & 93.0, 96.0, 98.0,100.0, 103.0, 106.0, 108.0, 111.0, 114.0, 118.0,
     &122.0,125.0,127.0,130.0, 133.0, 134.0, 137.0, 139.0, 141.0, 146.0,
     &146.0,149.0,153.0,156.0, 159.0, 161.0, 165.0, 166.0, 169.0, 172.0,
     &175.0,178.0,181.0,183.0, 186.0, 188.0, 192.0, 194.0, 197.0, 200.0,
     &204.0,206.0,209.0,  0.0,   0.0,   0.0,   0.0, 226.0, 227.0, 232.0,
     &231.0,238.0/

C... Parameters for calculating OMEGA of Equation (1) in ref [1].
      DATA  C2/ 5*1.0, 0.60, 1.00, 1.00, 1.00, 1.00, 46*1.0/
      DATA  C3/ 5*1.0, 1.00, 1.80, 0.70, 3.00, 1.00, 46*1.0/
      DATA  C4/ 6*1.0, 0.95, 1.00, 0.65, 1.30, 1.00, 45*1.0/
      DATA  C5/ 7*1.0, 0.20, 1.00, 0.70, 1.70, 1.00, 44*1.0/

C... Parameters for calculating sigma_0 of Equation (1) in ref [1].
      DATA  QC/ 13.0/, QL / 13.0/ , B / 1.15/

C... Parameters for calculating P of Equation (1) in ref [1].
      DATA  PC/ 0.0/, PL / 0.16/

C... Parameters for calculating R of Equation (1) in ref [1].
      DATA  RC/ 1.80/, RL / 10.7/ , RU/ 0.25/

C... Parameters for calculating S of Equation (1) in ref [1].
      DATA  S / 0.54/, SE / 1.4/, C  / 0.32/

C... Variable T of Equation (1) in ref [1].
      DATA  T/ 0.003/

C... Parameter eta of Table 1A of ref [1].
      DATA ET / 1.15, 1.15, 0.9, 0.8/

      END
C
C
      BLOCK DATA Y2BLOK

      COMMON /BT/ QM,C5,C6,C7,C8,C9,CD,D1,D2,D3,D4,D5,D6,PC,PL,PU,PG,PH,
     &               S,SE,C,RC,RL,RU,T,ET

C... The variable CJ holds the values for the parameter OMEGA of [1].
      DIMENSION  ET(4), QM(7)
      DIMENSION  C5(56), C6(56), C7(56), C8(56), C9(56), CD(56)
```

```
      DIMENSION  D1(56), D2(56), D3(56), D4(56), D5(56), D6(56)
      DATA  C5/ 7*1.0, 0.20, 1.00, 0.70, 1.70, 1.00, 44*1.0/
      DATA  C6/ 9*1.0, 0.60, 46*1.0/
      DATA  C7/12*1.0, 0.39, 1.00, 2.00, 41*1.0/
      DATA  C8/14*1.0, 1.20, 1.00, 1.00, 39*1.0/, C9/17*1.0,1.00,38*1.0/
      DATA  CD/18*1.0, 0.60, 1.00, 0.83, 35*1.0/, D1/21*1.0,1.20,34*1.0/
      DATA  D2,D3,D4,D5,D6/56*1.0, 56*1.0, 56*1.0, 56*1.0, 56*1.0/

C... Parameters for determining sigma0 in Table 1B of ref [1].
      DATA QM / 27.6, 0.66667, 1.0, 1.0, 0.3, 0.05, 0.5/

C... Parameters for determining P of [1].
      DATA  PC,PL,PU/0.075, 2.60, 0.50/, PG,PH /0.77, 0.66667/

C... Parameters for determining S of [1].
      DATA  S, SE, C/0.502, 1.4, 0.26/

C... Parameters for determining R of [1].
      DATA  RC,RL,RU/1.60, 10.2, 0.26/

C... The variable T in [1].
      DATA T/ 0.0005/

C... Parameter eta as in Table 1B of ref [1].
      DATA ET / 1.15, 1.15, 0.9, 0.8/

      END
C
C
      BLOCK DATA Y3BLOK

      COMMON /CT/ BA,QT,P0,P1,P2,P3,P4,P5,R0,R1,R2,S0,S1,T,ET

      DIMENSION   BA(92), QT(7), ET(4)
      DATA  BA / 16 * 0.0,  36.0,  38.0,  40.0,  44.0,
     & 45.0, 48.0, 50.0, 52.0,  55.0,  55.7,  58.5,  61.0,  64.0,  67.0,
     & 70.0, 73.0, 75.0, 78.0,  80.0,  82.0,  84.0,  86.0,  89.0,  92.0,
     & 93.0, 96.0, 98.0,100.0, 103.0, 106.0, 108.0, 111.0, 114.0, 118.0,
     &122.0,125.0,127.0,130.0, 133.0, 134.0, 137.0, 139.0, 141.0, 146.0,
     &146.0,149.0,153.0,156.0, 159.0, 161.0, 165.0, 166.0, 169.0, 172.0,
     &175.0,178.0,181.0,183.0, 186.0, 188.0, 192.0, 194.0, 197.0, 200.0,
     &204.0,206.0,209.0,   0.0,   0.0,   0.0,   0.0, 226.0, 227.0, 232.0,
     &231.0,238.0/

C... Parameters used in calculating sigma0 in Table 1C of ref [1].
      DATA QT/ 27.6, 0.6667, 1.0, 1.0, 0.3, 0.05, 0.5/

C... Parameters used in calculating parameter P in Table 1C of ref [1].
      DATA P0,P1,P2,P3,P4,P5 /20.0, 0.77, 1.98, 0.92, 0.77, 0.6667/

C... Parameters used in calculating parameter R in Table 1C of ref [1].
      DATA R0,R1,R2 /10.2, 0.26, 1.60/

C... Parameters used in calculating parameter S in Table 1C of ref [1].
      DATA S0,S1 /0.502, 0.08/

C... Parameters T in Table 1C of ref [1].
      DATA T /0.0005/

C... Values for eta in Table 1C of ref [1].
```

```
      DATA ET /1.25, 0.9, 1.0, 0.85/

      END
C
C
      BLOCK DATA Y4BLOK

      COMMON /DT/ BA,S0,S1,T3,R0,R1,R2,R3,P0,P1,P2,P3,E0,E1,C1,C2,C3,C4,
     &           D1,D2,T2,ET

C... BA is an array representing the average atomic number for all stable
C... isotopes of a given element.
      DIMENSION  BA(92), ET(4)
      DATA  BA / 16 * 0.0,  36.0,  38.0,  40.0,  44.0,
     & 45.0, 48.0, 50.0, 52.0,  55.0,  55.7,  58.5,  61.0,  64.0,  67.0,
     & 70.0, 73.0, 75.0, 78.0,  80.0,  82.0,  84.0,  86.0,  89.0,  92.0,
     & 93.0, 96.0, 98.0,100.0, 103.0, 106.0, 108.0, 111.0, 114.0, 118.0,
     &122.0,125.0,127.0,130.0, 133.0, 134.0, 137.0, 139.0, 141.0, 146.0,
     &146.0,149.0,153.0,156.0, 159.0, 161.0, 165.0, 166.0, 169.0, 172.0,
     &175.0,178.0,181.0,183.0, 186.0, 188.0, 192.0, 194.0, 197.0, 200.0,
     &204.0,206.0,209.0,  0.0,   0.0,   0.0,   0.0, 226.0, 227.0, 232.0,
     &231.0,238./

C... Parameters for calculating S, ?????? and R as in Table 1D of ref [1].
      DATA S0,S1,T3/0.482,0.07,3.0E-7/, R0,R1,R2,R3/11.8,0.45,1.29,0.15/

      DATA  P0,P1,P2,P3/1.98, 0.92, 20.0, 0.77/, E0,E1/20.3, 1.169/

C... Parameters for calculating sigma0 in Table 1D of ref [1] and Delta A_c
C... in Equation (2) of ref [1].
      DATA  C1,C2,C3,C4/144.0, 0.367, 0.3, 0.7/, D1,D2/0.0365, 1.23/

C... T in Table 1D of ref [1].
      DATA  T2/ 2.8E-4/

C... Parameter eta in Table 1D of ref [1].
      DATA ET / 1.25, 0.9, 1.0, 0.85/

      END
C
C
C
C
      SUBROUTINE YIELD1 (IZ, IA, JZ, JA, EJ, QJ)
C.......................................................................
C.......................................................................
C...
C... This subroutine is for the case where the incoming nucleus is    ...
C... at least as large is lithium and the secondary nucleus is        ...
C... between helium and beryllium, i.e.,                              ...
C...         5 .LE. IZ .LE. 16   .AND.   5 .LE. JZ .LE IZ             ...
C...                                                                  ...
C...                                                                  ...
C.......................................................................
C.......................................................................

C... Get parameters as in Table 1! of ref [1].
      EXTERNAL Y1BLOK

      COMMON /FS/ QR,QE,QF,QH,FE,FF,FA,FZ,PJ,PN,GA,ANZJ,AA,AE,AC,EC
      COMMON /QG/ QI,G1,G2,G3,G4
```

```
C... Commented out by Mark Mattson, April 19, 1996.
C     COMMON /ST/ ST,SS,T
      COMMON /AT/ BA,C2,C3,C4,C5,QC,QL,B,PC,PL,RC,RL,RU,S,SE,C,T,ET

      DIMENSION BA(92), CJ(56,26), C2(56), C3(56), C4(56), C5(56), ET(4)

C... Parameters for calculating OMEGA of Equation (1) in ref [1].
      EQUIVALENCE(CJ(1, 2),C2),(CJ(1, 3),C3),(CJ(1, 4),C4),(CJ(1, 5),C5)

      QR = 0.0
      QE = 0.0
      QF = 0.0
      QH = 0.0
      AE = 0.0
      AC = 0.0
      FE = 1.0   ! Initial value of f(E) of Equation (1) of ref [1].
      FF = 1.0
      FA = 1.0   ! Initial value of f(A) of Equation (2) of ref [1].
      FZ = 1.0
      PN = 1.0
      GA = 1.0

C... REAL*4 values for the integers describing atomic no. and atomic wgt.
      AI = IA
      ZI = IZ
      AJ = JA
      ZJ = JZ

C... Difference in atomic wgts.
      AA = AI - AJ

C... REAL*4 Number of neutrons in secondary nucleus.
      AN = AJ - ZJ

C... Ratio of neutrons to protons in secondary nucleus.
      ANZJ = AN/ZJ

C... Determination of the nuclear structure function, OMEGA, of Equation (1)
C... in ref [1].
      PJ = CJ(JA,JZ)

C... Determination of ratio (N/Z)* as on p. 349 of ref [2]; used in
C... calculation of f(A) of equation (1) of ref [1].
      AM = BA(IZ)
      IF (AM.EQ.0) AM = IA
      CN = 0.3*(AI - AM)/ZI

C... Change in the number of neutrons between target and secondary nuclei.
      KN = (IA - IZ) - (JA - JZ)

C... Change in the number of protons between target and secondary nuclei,
C... including initial proton.
      JP = IZ - JZ + 1

C... Integer number of neutrons in secondary nucleus.
      JN = JA - JZ

C... Determination of eta of Equation (1)
C...
      MN = JN .AND. 1
```

```
      MZ = JZ .AND. 1
C     PN = 1.15
C     IF(MZ.EQ.1 .AND. MN.LE.1) PN = 0.9 - 0.1*MN
      IF(MZ.EQ.0 .AND. MN.EQ.0) PN = ET(1)
      IF(MZ.EQ.0 .AND. MN.EQ.1) PN = ET(2)
      IF(MZ.EQ.1 .AND. MN.EQ.0) PN = ET(3)
      IF(MZ.EQ.1 .AND. MN.EQ.1) PN = ET(4)

C... Determination of cutoff energy, E0 (either 1250 MeV or from Equation (3)
C... of ref [1].).
      EC = 68.7*AI**0.866
      IF (EC.LT.1250.0) EC = 1250.0
      EI = EJ
      IF (EI.GT.EC) EI = EC

C... Calculation of correction factor to H(E) in Equation (25) on page 358
C... of ref [2].
      H3 = 1.0
      IF (EI.LT.80.0) H3 = 1. - EXPF(-(EI/25.)**4)

C... If the secondary nucleus is higher than helium or if the change in the
C... number of neutrons is greater than 1, it will be handled later.
      IF(JP.GT.2 .OR. KN.GT.1) GO TO  3

C... Calculation of correction factor as described on p. 358 of ref [2];
C... CX .NE. 1 is for light secondary nuclei.
      CX = 1.0
      IF (JZ.EQ.4 .AND. JA.EQ.9) CX = CJ(JA,JZ)/PN

C... Calculation of correction factor for light nuclei as described on
C... p. 876 of ref [3].
      IF (AI/ZI.GT.2.0) CT = (ZI - 2.0)/((AI - ZI) - 2.0)
      IF (ZI.GT.5 .OR. CT.GT.1.0 .OR. (AI-ZI).LE.2.0) CT = 1.0

 1    IF (JP.GT.1) GO TO  2  ! Case of > 1 proton in secondary nucleus.
      IF (KN.GT.1) GO TO  3  ! Case of > 1 neutron created in reaction.

C... Cross section for (p,pn) as shown in Equation (24) of ref [2] on p. 357.
      IF (AI.LE.40.0) QN = 24.0*(1.0 + 0.01*AI)
      IF (AI.GT.40.0) QN = 1.02*(AI - 7.0)
      IF (AI.GE.63.0) QN = 57.0

C... Calculation of H(E) as in Equation (25) on p. 358 of ref [2].
      IF (EI.LT.2500.0)
     &      FE = (1.0 + 2.1*EXP(-(EI/100.0)**2) + 0.4*EXP(-EI/350.0))*H3

C... Calculation of cross section for (p,pn) reaction.
      QH = QN*CX
      QJ = QH
      IF (EI.LT.EC) QJ = QH*FE
      RETURN

 2    IF (KN.NE.0) GO TO 3  ! Case for no neutrons created in reaction.
C... Calculation of cross section for (p,2p) reaction as shown in Equations
C... (26) and (27) on p. 358 of ref [2].
C... This is the calculation of H(E) as in Eq. (27) on p. 358 of ref [2].
      IF (EJ.LT.2500.0) FE = (1.0 - EXPF(-(EJ/230.)**2) +
     &      2.2*EXPF(-EJ/75.0) + 0.33*EXPF(-((EJ-900.0)/500.0)**2))*H3

C... 21.0 is the value of sigma(E0) as shown in Eq. (26) of ref [2].
```

```
      QH = 21.0*CX*CT
      QJ = QH*FE
      RETURN

C... If the atomic number is greater than 29, then the cross-section needs
C... to be modified by f(A) and f(E) as outlined in Table 1 on page 340 of
C... ref [2] (FE and FA were previously set equal to 1).
 3    IF (IZ.LT.29) GO TO  5

C... Calculation of f(E) of Equation (1) of ref [1].
      IF (JZ.NE. 4) FE = (EI/EC)**(0.4*ZJ)     ! Helium and Lithium nuclei
      IF (JZ.EQ. 4) FE = (EI/EC)**1.8          ! Beryllium nuclei

C... Calculation of f(A) in Equation (1) or ref [2]; Eq. (10) on p. 351 of
C... ref [2].
      FA = EXPF(0.01*(AI - 56.0)*(AN/ZJ - CN - 0.45))
      IF (FA.LE.1.0) FA = 1.

C...
      EI = EC
      ZI = 29.0
      AI = 63.0

C... Calculation of enhancement factor xi of Equation (1) of ref [1];
C... see Table 2 of ref [3].
 5    AT = IA
      IF (IA.GE.104) AT = 104
      IF (IA.GE.64) GO TO 7
      IF (IA.GT.34) GO TO 6

C... Situation described in Table 3 of ref [1].
      IF (IA.GE.14 .AND. JA.EQ.6 .AND. JZ.EQ.2) FZ = 1.0 + 0.1*(IA - 14)
      GO TO 8

C... Enhancement factor xi when target nucleus has 34 .LE. IA .LE. 63
 6    IF (JA.EQ.6 .AND. JZ.EQ.2) FZ = 3.0*(1.0 + 0.02*(IA - 34))
      IF (JA.EQ.6 .AND. JZ.EQ.3) FZ = 1.0 + 0.02*(IA - 34)
      IF (JA.EQ.7 .AND. JZ.LE.4) FZ = 1.0 + 0.01*(IA - 34)
      GO TO 8

C... Enhancement factor xi when target nucleus has 64 .LE. IA .LE. 104
 7    IF (JA.EQ.6  .AND. JZ.EQ.2) FZ = 4.8 + 0.0450*(AT - 64.0)
      IF (JA.EQ.6  .AND. JZ.EQ.3) FZ = 1.6 + 0.0150*(AT - 64.0)
      IF (JA.EQ.7  .AND. JZ.EQ.3) FZ = 1.3 + 0.0150*(AT - 64.0)
      IF (JA.EQ.7  .AND. JZ.EQ.4) FZ = 1.3 + 0.0105*(AT - 64.0)
      IF (JA.EQ.8  .AND. JZ.EQ.3) FZ = 1.0 + 0.0225*(AT - 64.0)
      IF (JA.EQ.9  .AND. JZ.EQ.3) FZ = 1.0 + 0.0125*(AT - 64.0)
      IF (JA.EQ.9  .AND. JZ.EQ.4) FZ = 1.0 + 0.0100*(AT - 64.0)
      IF (JA.GE.10 .AND. JZ.LE.5) FZ = 1.0 + 0.0050*(AT - 64.0)

 8    DA = AI - AJ  ! Change in atomic weights of target and secondary nuclei.
      AA = DA

C... Calculation of delta A_c as in Equation (2) of ref [1].
      AE = 31.5 + 0.052*(AI - 36.0)*(ALOG(EI) - 3.17)

C... Determination of delta A in Equation (1) of ref [1].
      IF (AA.GT.AE) DA = AE

C... Determination of sigma0 in Equation (1) of ref [1].
```

```
      Q1 = QC
      IF (EI.LT.1250.0) Q1 = QL*EXPF(B*(1.0 - 0.0008*EI))

C... Determination of P in Equation (1) of ref [1].
      PE = PC
      IF (EI.LT.1250.0) PE = PL*(1. - 0.0008*EI)

C... Determination of R in Equation (1) of ref [1].
      RE = RC
      IF (EI.LT.1250.0) RE = RL/EI**RU
      IF(IZ.GT.20 .AND. EI.LT.1250.0) RE = 1.8

C... Correction factor for sigma0 (f3 in Table 1A of ref [1]).
      IF(EI.LT.1250.0 .AND. IZ.GE.21)
     &                     Q1 = Q1*2.0*EXPF(-((EI - 650.0)/720.0)**2)

C... Determination of S in Equation (1) of ref [1].
      IF (AI/ZI.GE.2.0) SS = S - C*(AI/ZI - 2.0)**SE
      IF (AI/ZI.LT.2.0) SS = S + C*(2.0 - AI/ZI)**SE  ! See ref [3]

C... Determination of portion of exponent in Equation (1) of ref [1]
C... within absolute value symbol.
      ST = (ZJ - (SS - T*AJ)*AJ)
      ZA = (ZJ - (SS - T*AJ)*AJ)**2

C... Calculation of cross section.
      QR = Q1*EXPF(-PE*DA - RE*ZA)*FA*FZ*CJ(JA,JZ)
      QH = QC*EXPF(-RC*ZA)*FA*FZ*CJ(JA,JZ)
      QE = QH*FE
      IF (IZ.LT.29) QJ = QR
      IF (IZ.GE.29) QJ = QE

C... Enhancement factor for (p,3p) reactions, as shown on p. 876 of ref [3].
      IF (JP.EQ.3 .AND. KN.EQ.0) QJ = 1.5*QR

C     FOLLOWING REVISIONS ARE ADDED 12/27/78
C... From ref [3], the cross section QJ is to be modified by a correction
C... factor.
      IF (IZ.LE.20) RETURN
      QI = QJ

C... Determination of E' as shown below Equation (3) of ref [3].
      EX = 68.7*56.0**0.866*EJ/EC
      AC = 31.5 + 0.045*(AI - 36.0)*(ALOG(AI) + 1.23)

C... Equation (3) of ref [3].
      G1 = 1.0 - 0.6*(1.0 - EXPF(-(EJ/1000.0)**2))*EXPF(-(EJ/2000.0)**2)
     &        + 0.2*(1.0 - EXPF(-(EJ/3000.0)**2))
      GX = 1.0 - 0.6*(1.0 - EXPF(-(EX/1000.0)**2))*EXPF(-(EX/2000.0)**2)
     &        + 0.2*(1.0 - EXPF(-(EX/3000.0)**2))

C... Calculation of f2 in Equation (9) of ref [3].
      G2 = 1.0 - 0.4*(1.0 - EXPF(-(EX/2000.0)**2))*
     &       EXPF(-((EX-1800.0)/1800.0)**2) +
     &       0.17*(1.0 - EXPF(-(EX/2000.0)**2))

C... Determination of f1 for the conditions described in Eq. (4) of ref [3].
      IF (EJ.GT.2500.) G1 = GX
      IF (EJ.GT.EC .AND. EJ.LT.2500.0) G1 = SQRT(G1*GX)
```

```
C... Modification of the cross section.
      IF (IZ.GE.21 .AND. IZ.LE.28) QJ = QR*G1
      IF (IZ.GT.28 .AND. AA.GT.AC) QJ = QE*G2

      RETURN
      END
C
C
C
C
      SUBROUTINE YIELD2 (IZ, IA, JZ, JA, EJ, QJ)
C.....................................................................
C.....................................................................
C...                                                               ...
C... This subroutine is for the case where the incoming nucleus is ...
C... for elements between boron and sulphur and the secondary nucleus...
C... is between boron and the incoming nucleus, i.e.,              ...
C...        5 .LE. IZ .LE. 16   .AND.   5 .LE. JZ .LE IZ           ...
C...                                                               ...
C.....................................................................
C.....................................................................

C... Get parameters is described in Table 1B of ref [1].
      EXTERNAL Y2BLOK

      COMMON /BT/ QM,C5,C6,C7,C8,C9,CD,D1,D2,D3,D4,D5,D6,PC,PL,PU,PG,PH,
     &            S,SE,C,RC,RL,RU,T,ET
      COMMON /FS/ QR,QE,QF,QH,FE,FF,FA,FZ,PJ,PN,GA,ANZJ,AA,AE,AC,EC
C... Commented out by Mark Mattson, April 19, 1996.
C     COMMON /ST/ ST,SS,T

C... The variable CJ holds the values for the parameter OMEGA of [1].
      DIMENSION  CJ(56,26), ET(4), QM(7)
      DIMENSION  C5(56), C6(56), C7(56), C8(56), C9(56), CD(56)
      DIMENSION  D1(56), D2(56), D3(56), D4(56), D5(56), D6(56)
C     EQUIVALENCE(CJ(1, 5),C5),(CJ(1, 6),C6),(CJ(1, 7),C7),(CJ(1, 8),C8)
C     EQUIVALENCE(CJ(1, 9),C9),(CJ(1,10),CD),(CJ(1,11),D1),(CJ(1,12),D2)
C     EQUIVALENCE(CJ(1,13),D3),(CJ(1,14),D4),(CJ(1,15),D5),(CJ(1,16),D6)

      DO I=1,56
        CJ(I,5)  = C5(I)
        CJ(I,6)  = C6(I)
        CJ(I,7)  = C7(I)
        CJ(I,8)  = C8(I)
        CJ(I,9)  = C9(I)
        CJ(I,10)  = CD(I)
        CJ(I,11)  = D1(I)
        CJ(I,12)  = D2(I)
        CJ(I,13)  = D3(I)
        CJ(I,14)  = D4(I)
        CJ(I,15)  = D5(I)
        CJ(I,16)  = D6(I)
      ENDDO

      QR = 0.0
      QE = 0.0
      QF = 0.0
      QH = 0.0
      AE = 0.0
      AC = 0.0
```

```
      FE = 1.0
      FF = 1.0
      FA = 1.0
      FZ = 1.0
      GA = 1.0
      AI = IA       ! Real number for the atomic weight of the incoming nucleus
      ZI = IZ       !  "     "     "   "    "    number "   "       "       "
      AJ = JA       !  "     "     "   "    "    weight "   "  secondary    "
      ZJ = JZ       !  "     "     "   "    "    number "   "       "       "
      AA = AI - AJ ! Difference in atomic weights of nuclei
      AN = AJ - ZJ ! Number of neutrons in the secondary nucleus
      ANZJ = AN/ZJ ! Ratio of neutrons to protons in secondary nucleus

C... Determination of OMEGA in ref [1].
      PJ = CJ(JA,JZ)

      JN = JA - JZ ! Integer for number of neutrons in secondary nucleus
      MN = JN.AND.1 ! Is the # of neutrons even or odd?
      MZ = JZ.AND.1 ! Is the # of protons even or odd?
      CX = 1.

C... The variable PN is eta as in ref [1].
C     PN = 1.15
C     IF(MZ.EQ.1 .AND. MN.LE.1) PN = 0.9 - 0.1*MN
      IF(MZ.EQ.0 .AND. MN .EQ.0) PN = ET(1)
      IF(MZ.EQ.0 .AND. MN .EQ.1) PN = ET(2)
      IF(MZ.EQ.1 .AND. MN .EQ.0) PN = ET(3)
      IF(MZ.EQ.1 .AND. MN .EQ.1) PN = ET(4)

C... Determination of E0 ("critical energy") as in ref [1];
C... 1250 MeV is the lower bound.
      EC = 68.7*AI**.866
      IF (EC.LT.1250.0) EC = 1250.0

C...
      EI = EJ
      IF (EI.GT.EC) EI = EC
      H3 = 1.0
      IF (EI.LT.80.0) H3 = 1.0 - EXPF(-(EI/25.)**4)

      KN = (IA - IZ) - (JA - JZ)  ! Difference in number of n's
      JP = IZ - JZ + 1            !     "      "    "    "  p's (inc. 1st)

      IF((JZ.EQ.7 .AND. JA.EQ.13) .OR. (JZ.EQ.10 .AND. JA.EQ.19))
     &      CX = CJ(JA,JZ)

C... Correction factor as described on p. 876 of ref [3].
      CT = (ZI - 2.0)/((AI - ZI) - 2.0)
      IF (ZI.GT.5.0 .OR. CT.GT.1.0) CT = 1.0

      IF (JP.GT.1) GO TO  2
 1    IF (KN.NE.1) GO TO  3

C... Factors for (p, pn) reactions.
C... Calculation of H(E) as in Eq. (25) on p. 358 of ref [2].
      IF (EI.LT.2500.0)
     &   FE = (1.0 + 2.1*EXPF(-(EI/100.0)**2) + 0.4*EXPF(-EI/350.0))*H3

C... Equation (24) on p. 357 of ref [2].
      QH = 24.0*(1.0 + 0.01*AI)*CX
```

```
      QJ = QH
      IF (EI.LT.EC) QJ = QH*FE
      GO TO 10

 2    IF(JP.GT.2 .OR. KN.NE.0) GO TO 3
C... Factors for (p, 2p) reactions.
C... Calculation of H(E) as in Eq. (27) on p. 358 of ref [2].
      IF (EJ.LT.2500.0)
     &      FE = (1.0 - EXPF(-(EJ/230.0)**2) + 2.2*EXPF(-EJ/75.0)
     &    + 0.33*EXPF(-((EJ-900.0)/500.0)**2))*H3

C... Eq. (26) on p. 358 of ref [2] multiplied by correction factors.
      QH = 21.0*CX*CT
      QJ = QH*FE
      GO TO 10

C... Determination of a portion of sigma0 as in ref [1].
 3    F1 = QM(4) - QM(5)*ALOG(AI*QM(6))

C... Determination of the parameter P of [1].
      PE = PC
      IF (EC.GT.1250.0) PE = PG/AI**PH
      IF (EI.LT.EC) PE = PL/EI**PU

C... Determination of the parameter R of [1].
      RE = RC
      IF (EI.LT.1250.0) RE = RL/EI**RU

C... Determination of the parameter S of [1].
      IF (AI/ZI.GE.2.0) SS = S - C*(AI/ZI - 2.0)**SE
      IF (AI/ZI.LT.2.0) SS = S + C*(2.0 - AI/ZI)**SE

C... Determination of the value in the exponential in Equation (1) in ref [1].
      ST = (ZJ - (SS - T*AJ)*AJ)
      ZA = (ZJ - (SS - T*AJ)*AJ)**2

C... Determination of sigma0 as in ref [1].
      Q1 = QM(1)*(AI**QM(2) - QM(3))*F1*PE*RE**QM(7)/(1.0-EXPF(-PE*AI))
      QC = QM(1)*(AI**QM(2) - QM(3))*F1*PC*RC**QM(7)/(1.0-EXPF(-PC*AI))

C... Determination of the cross section.
      QR = Q1*EXPF(-PE*(AI - AJ))*EXPF(-RE*ZA)*CJ(JA,JZ)*PN
      QJ = QR
      QH = QC*EXPF(-PC*(AI - AJ))*EXPF(-RC*ZA)*CJ(JA,JZ)*PN
      IF(JP.LT.3 .OR. KN.NE.0) GO TO 10

C...
      QJ = QR*AMIN1(.0022*AJ*AJ,1.)
      IF (JP.NE.4) GO TO 10
      IF (QJ.GT.0.5) QJ = 0.5

C... Determination of the enhancement factor, xi, as in ref [1].
 10   IF (IZ.GE.14 .AND. EI.GE.500.0) QJ=QJ*(1.0 + 0.12*(IZ - 13))
      IF (IZ.GE.14 .AND. EI.GE.200.0 .AND. EI.LT.500.0)
     &    QJ = QJ*(1.0 + 0.12*(IZ-13)*EXP(-((EI-500.0)/350.0)**2))
      IF (JP.EQ.2 .AND. KN.EQ.1 .AND. IZ.GE.12) QJ = QJ*1.7
      IF ((AJ/ZJ).GT.1.8 .OR. IZ.GT.10) RETURN
      IF (JP.NE.1 .OR.  KN.NE.2) QJ = QJ*0.3
      IF (JP.EQ.1 .AND. KN.EQ.2) QJ = QJ*0.5
```

```
      RETURN
      END
C
C
C
C
      SUBROUTINE YIELD3(IZ, IA, JZ, JA, EJ, QJ)
C.........................................................................
C.........................................................................
C...                                                                   ...
C... This subroutine is for the case where the incoming nucleus is    ...
C... for elements between chlorine and calcium and the secondary      ...
C... nucleus is between boron and the incoming nucleus, i.e.,          ...
C...        17 .LE. IZ .LE. 20   .AND.   5 .LE. JZ .LE. IZ             ...
C...                                                                   ...
C.........................................................................
C.........................................................................

C... Get parameters as in Table 1C of ref [1].
      EXTERNAL Y3BLOK

      COMMON /CT/ BA,QT,P0,P1,P2,P3,P4,P5,R0,R1,R2,S0,S1,T,ET
      COMMON /FS/ QR,QE,QF,QH,FE,FF,FA,FZ,CJ,PN,GA,ANZJ,AA,AE,AC,EC
C... Commented out by Mark Mattson, April 19, 1996.
C     COMMON /ST/ ST,S,T
      DIMENSION   BA(92), QT(7), ET(4)

C... REAL*4 versions of the atomic weights and numbers of the target and
C... secondary nucleus.
        AI = IA
        ZI = IZ
        AJ = JA
        ZJ = JZ

        AM = BA(IZ)
        IF (AM.EQ.0.0) AM = IA

C... Calculation of E0 as in Equation (3) of ref [1].
        EC = 68.7*AI**0.866
        IF (EC.LT.1250.0) EC = 1250.0
        EI = EJ
        IF (EI.GT.EC) EI = EC

        QR = 0.0
        QE = 0.0
        QF = 0.0
        QH = 0.0
        AE = 0.0
        AC = 0.0
        DX = 0
        FE = 1.0
        FF = 1.0
        FA = 1.0
        FZ = 1.0
        GA = 1.0
        CJ = 1.0
        AA = AI - AJ

C... Calculation of last portion of H(E) in Equation (25) of ref [2].
        H3 = 1.0 - EXPF(-(AMIN1(EI,80.0)/25.0)**4)
```

```
C... Calculation of H4 in Equation (29d) of ref [2].
        H4 = AMIN1(AMAX1((400.0/EI-2.2),1.0),2.0)

C... Number of neutrons in secondary nucleus and ratio of neutrons to protons.
        AN = AJ - ZJ
        ANZJ = AN/ZJ

C... Calculation of eta as in Table 1C of ref [1].
        JN = JA - JZ
        MN = JN.AND.1
        MZ = JZ.AND.1
C       PN = 1.0
C       IF (MN+MZ.EQ.2) PN = 0.85
C       IF (MN+MZ.EQ.0) PN = 1.25
C       IF (MN-MZ.EQ.1) PN = 0.90
        IF(MZ.EQ.0 .AND. MN.EQ.0) PN = ET(1)
        IF(MZ.EQ.0 .AND. MN.EQ.1) PN = ET(2)
        IF(MZ.EQ.1 .AND. MN.EQ.0) PN = ET(3)
        IF(MZ.EQ.1 .AND. MN.EQ.1) PN = ET(4)

C... Total change in number of protons.
        JP = IZ - JZ + 1

C... Total change in number of neutrons.
        KN = IA - IZ - JN

        IF(JP.GT.2 .OR. KN.GT.3 .OR. IA.LT.35) GO TO 14
        IF (JP.EQ.2) GO TO 2

C... Case for where the change in the number of protons is 1, there are no
C... more than two additional neutrons and the target nucleus has an atomic
C... weight greater than 35.
C... Calculation of H(E) as in Equation (25) of ref [2].
 1      IF (EI.LT.2500.0)
     &      FE =(1.0 + 2.1/EXPF((EI/100.0)**2) + 0.4*EXPF(-EI/350.0))*H3

C... Calculation of cross section if the weight of the target nucleus is
C... less than or equal to 40, as in Equation (24) of ref [2].
        QH = 24.0*(1.0 + 0.01*AI)
        QJ = QH

C... Determination of variable d as described below Equation (23) in ref [2].
        DX = 3.0
        IF (KN.GT.1) DX = 15.0

        IF (EI.LT.EC) QJ = QH*FE

C... Temporarily finished for (p,pn) reactions.
        IF (KN.EQ.1) GO TO 5

C... (p,pxn) where x .GE. 2. (p. 359 of ref [2]).
        XN = KN
        XA = 1.17
        IF (IZ.LE.30) XA=1.6

C... Calculation of cross section as in Equation (28) of ref [2].
        QH = QH*EXPF(1.0 - XN**(XA - 0.0048*AI))
        QJ = QH
        IF (EI.GE.EC) GO TO 5
```

```
C... Temporarily finished with (p,pxn)
        GO TO 4

C... (p,2pxn) reactions.
C... Calculation of f(e) as in Equation (27) of ref [2].
 2      IF (EJ.LT.2500.0)
     &      FE = (1.0 - EXPF(-(EJ/230.0)**2) + 2.2/EXPF(EJ/75.0)
     &    + 0.33/EXPF(((EJ-900.0)/500.0)**2))*H3

C... Calculation of cross section as in Equation (26) of ref [2].
        QH = 21.0
        QJ = QH
        QJ = QH * FE

C... Determination of variable d as described below Equation (23) of ref [2].
        IF (KN.EQ.0) DX = -3.0
        IF (KN.EQ.1) DX = -1.0

C... Temporarily finished with (p,2p) reactions.
        IF (KN.EQ.0) GO TO 5

C... Temporarily finished with (p,2pxn) where n > 2.
        IF (KN.GT.2) GO TO 14

C... Cross section for (p,2pxn), x .GE. 1 as in Equation (31) of ref [2].
        QH = 17.0
        QJ = QH

        IF (EI.GE.EC) GO TO 5
C... Calculation in change in atomic weights.
 4      KA = IA - JA
        IF (KA.GE.8) GO TO 5

C... Calculation of (1 + H1)*H3*H4 as almost in Equation (29) of ref [2].
C...
        FE = (1.0 + 1.9/EXP((AA/7.9)**2 + (EI/420.0)**1.4))*H3*H4
        QJ = QH*FE

C... Calculation of Y(IA,IZ) as in Equation (22) in ref [2].
 5      DD = DX*(AI - AM)/ZI
        YA = EXPF(DD)
        IF (DD.GT.0) YA = 2.0 - 1.0/YA
        IF (IA.LT.35 .OR. IA.GT.209) YA = 1.0
        IF (IA.LT.70 .AND. JP.EQ.3) YA = 1.0

C... Calculation of cross section.
        QH = YA*QH
        QJ = YA*QJ
        RETURN

C... Calculations for when the change in the number of protons is greater
C... than 2, the change in the number of neutrons is greater than 3 or the
C... atomic weight of the target nucleus is less than 35.
C... Calculation of change in atomic weights.
 14     DA = AI - AJ

C... Determination of OMEGA for what in this region is a special case (see
C... Table 2 of ref [1]).
        IF(JZ.EQ.7 .AND. JA.EQ.13) CJ = 0.39
```

```
C... Calculation of portion of sigma0 as in Table 1C of ref [1].
        F1 = QT(4) - QT(5)*ALOG(AI*QT(6))

C... Calculation of f2' as in Equation (6) of ref [1].
        IF (EI.GE.600.0) F2 = 1.0
        IF (EI.LT.600.0) F2 = EXPF (0.90 - 0.0015*EI)

        IF (EI.GE.EC) GO TO 15
C... Determination of P in Table 1C in ref [1] for low energy.
        PE = P0/EI**P1
 15     IF (EI.LT.EC) GO TO 16

C... Determination of f2' and P in Table 1C of ref [1] for high energy.
        F2 = 1.00
        PE = P2/AI**P3

C... Determination of R in Table 1C of ref [1].
 16     IF (EI.LT.1250.0) RE = R0/EI**R1
        IF (EI.GE.1250.0) RE = R2

C... Calculation of sigma0 in Table 1C of ref [1].
        Q1 = QT(1)*(AI**QT(2) - QT(3))*F1*F2*PE*RE**QT(7)/
     &              (1.0 - EXPF(-PE*AI))

C... Calculation of S in Table 1C of ref [1].
        S = S0 - S1*ABS(AI/ZI - 2.0)

C... Determination of value in second exponent of Equation (1) of ref [1].
        ST = (ZJ - (S - 0.0005*AJ)*AJ)
        ZA = ABS(ZJ - (S - 0.0005*AJ)*AJ)**2

C... Calculation of cross section as in Equation (1) of ref [1].
        QJ = Q1*EXPF (-PE*DA-RE*ZA)
        QR = QJ*PN*CJ
        QJ = QR

C... Consideration of high energy case.
C... Calculation of P in Table 1C of ref [1].
        PH = P4/AI**P5

C... Determination of R in Table 1C of ref [1].
        RH = R2

C...
C... Calculation of sigma0 in Table 1C of ref [1].
        QC = QT(1)*(AI**QT(2) - QT(3))*F1*PH*RH**QT(7)/
     &              (1.0 - EXPF(-PH*AI))

C... Calculation of cross section as in Equation (1) of ref [1].
        QH = QC*EXPF(-PH*DA - RH*ZA)*PN*CJ
        IF(JP.NE.3 .OR. KN.GT.0 .OR. IA.LT.35) GO TO 20

C... Determination of correction factors in (p,3p) reactions.
        FF = 1.0
        FE = 1.0
        QJ = QH

 20     IF (IZ.LT.14 .OR. IZ.GT.19 .OR. EI.LT.200.0) RETURN
```

```
C... Correction factor xi.
C...
         IF (EI.GT.500.0) QJ = QJ*(1.0 + 0.12*(IZ-13))
         IF (EI.LE.500.0) QJ = QJ*(1.0 + 0.12*(IZ-13)*
     &       EXP(-((EI-500.0)/350.0)**2))

      RETURN
      END
C
C
C
C
      SUBROUTINE YIELD4(IZ, IA, JZ, JA, EJ, QJ)       ! see notes at end
C.....................................................................
C.....................................................................
C...                                                               ...
C... This subroutine is for the case where the incoming nucleus is   ...
C... for elements between scandium and uranium and the secondary     ...
C... nucleus is between boron and the incoming nucleus, i.e.,        ...
C...       21 .LE. IZ .LE. 92   .AND.   5 .LE. JZ .LE. IZ            ...
C...                                                               ...
C.....................................................................
C.....................................................................

C... Get parameters as in Table 1D of ref [1].
      EXTERNAL Y4BLOK

      COMMON /DT/ BA,S0,S1,T3,R0,R1,R2,R3,P0,P1,P2,P3,E0,E1,C1,C2,C3,C4,
     &            D1,D2,T2,ET
      COMMON /FS/ QR,QE,QF,QH,FE,FF,FA,FZ,CJ,PN,GA,ANZJ,AA,AE,AC,EC
      COMMON /Q4/ C0, F1, F2, F3, PE, DA, RJ, ZA, PH, YA, Q1, QM, HM
      COMMON /QG/ QI, G1, G2, G3, G4
C... Commented out by Mark Mattson, April 19, 1996.
C     COMMON /ST/ ST,S,T2

C... BA is an array representing the average atomic number for all stable
C... isotopes of a given element.
      DIMENSION   BA(92), ET(4)

C... REAL*4 versions of atomic weights and numbers of nuclei.
      AI = IA
      ZI = IZ
      AJ = JA
      ZJ = JZ

      AM = BA(IZ) + IA*(300/(INT(BA(IZ)) + 300))
      KM = 0

C...
      FP = 1.
      QJ = 0.0
      QR = 0.0
      QE = 0.0
      QF = 0.0
      QM = 0.0
      HM = 0.0
      QH = 0.0
      AC = 0.0
      AE = 0.0
      DM = 0.0
```

```
      DX = 0.0
      KFLAG = 0
      FE = 1.0
      FF = 1.0
      FM = 1.0
      FA = 1.0
      FZ = 1.0
      GA = 1.0
      CJ = 1.0
      PN = 1.0
      YA = 1.0

C... This subroutine doesn't consider reactions where the secondary nucleus
C... has more mass, protons or neutrons than the target nucleus.
      IF(IA.LE.JA .OR. IZ.LT.JZ .OR. (IA-JA).LT.(IZ-JZ)) RETURN

      AA = AI - AJ
      DA = AA
      AN = AJ - ZJ
      ANZJ = AN/ZJ
      ANZI = AI/ZI - 1.0
      JN = JA - JZ           ! Number of neutrons in product nucleus
      JP = IZ - JZ + 1       ! Change in number of protons
      CN = 0.3*(AI - AM)/ZI
      KN = (IA - IZ) - JN    ! Change in number of protons
      XN = KN

C... Calculation of eta in Table 1D of ref [1].
      MZ = JZ.AND.1
      MN = JN.AND.1
C     IF ((MN+MZ).EQ.2) PN = 0.85
C     IF ((MN+MZ).EQ.0) PN = 1.25
C     IF ((MN-MZ).EQ.1) PN = 0.90
      IF(MZ.EQ.0 .AND. MN.EQ.0) PN = ET(1)
      IF(MZ.EQ.0 .AND. MN.EQ.1) PN = ET(2)
      IF(MZ.EQ.1 .AND. MN.EQ.0) PN = ET(3)
      IF(MZ.EQ.1 .AND. MN.EQ.1) PN = ET(4)

C...
      PN = PN + (1.-PN)*(1.-EXPF(-((IA-100)/35.)**2))

C... Calculation of DELTA A_c as in Equation (2) of ref [1].
      AC = 31.5 + 0.045*(AI - 36.0)*(ALOG(AI) + 1.23)

      EI = EJ
      EC = E0*AI**E1
      IF (EC.GT.4000.0) EC = 4000.0
      IF (EC.LT.1250.0) EC = 1250.0
      IF (EI.GT.EC) EI = EC

C... Determination of maximum value for energy dependence of fission cross
C... section as described on p. 347-8 of ref [2].
      FMAX = 1800.0/EI
      IF (IZ.GE.84) FMAX = (1800.0/EI)**(6.56 - 0.067*ZI)
      IF (FMAX.GT.4.0) FMAX = 4.0
      IF (EI.GT.1800.0) FMAX = 1.0

C... Determination of OMEGA for certain situations.
      IF (IZ.LE.28 .AND. JZ.EQ.20 .AND. JA.EQ.19) CJ = 0.6
      IF (JZ.EQ.7 .AND. JA.EQ.13) CJ = 0.39
```

```
C...
C... Determination of parameter P of Table 1D of ref [1] for different
C... energies.
      P500 = P2/(500.0**P3)
      P1000 = P2/(1000.0**P3)
      P3000 = P2/(3000.0**P3)
      IF (JP.GE.6 .OR. IA.LT.35) GO TO 100

C...
      IF (JP.GE.4.AND.JP.LE.5.AND.KN.GE.KM.AND.EJ.LT.EC)  GO TO 100

C     PERIPHERAL

C... Calculation of H3 as in Equation (29c) of ref [2].
      H3 = 1.0 - EXPF(-(EI/(15.0 + 10.0*AA))**4)

C... Determination of H4 as in Equation (29d) of ref [2].
      H4 = 400.0/EI - 2.2
      IF (H4.LT.1.0) H4 = 1.0
      IF (H4.GT.2.0) H4 = 2.0

C... Calculation of H(E) as in Equation (29) of ref [2].
      HE = (1.0 + 1.9*EXPF(-(DA/7.9)**2 - (EI/420.0)**1.4)
     &       - (1.0 - EXPF(-(DA/12.0)**8))*EXPF(-(EI/420.0)**3))*H3*H4

      EC = 2500.0
      IF (JP.GE.3 .AND. JP.LE.5) GO TO 3
      IF (JP.EQ.2) GO TO 2

C... For (p,pn) reactions.
C... Calculation of H(E) as in Equation (25) of ref [2].
 1    IF (EI.LT.2500.0)
     &    FE = (1.0 + 2.1*EXPF(-(EI/100.0)**2) + 0.4*EXPF(-EI/350.0))*H3
      IF (KN.GT.1) FE = HE

C... Calculation of x_max as in Equation (30) of ref [2].
      KM = AI/20.0 + 1.5*(ABS(238.0 - AI)/167.0)**2.5 + 0.8
      IF (IA.LE.70) KM = 3.0 + AI/66.0

      IF (KN.GT.KM) GO TO 100

C... Calculation of cross section for (p,pn) as in Equation (24) of ref [2].
      IF (IA.LE.40) QA = 24.0*(1.0 + 0.01*AI)
      QH = QA
      IF (IA.GT.40) QA = 1.02*(AI - 7.0)
      QH = QA
      IF (IA.GE.63) QA = 57.0
      QH = QA

C... Determination of exponent of x in Equation (28) of ref [2] (see also
C... p. 876 of ref [3]).
      XA = 1.17
      IF (IZ.LE.30) XA = 1.5

C... Calculation of cross section for (p,pxn) as in Equation (28) of ref [2].
      IF (KN.NE.1) QH = QA*EXPF(1.0 - XN**(XA - 0.0048*AI))

C... Determination of d as described below Equation (23) of ref [2].
      IF (KN.EQ.1) DX = 3.0
      IF (KN.GT.1) DX = 15.0
```

```
C...
      QH = QH*(1. + 0.15*(IZ/80.)**2)

      GO TO 6

C... Calculation of H(E) as in Equation (27) of ref [2].
 2    FE = (1.0 - EXPF(-(EJ/230.0)**2) + 2.2*EXPF(-EJ/75.0)
     &    + 0.33*EXPF(-((EJ-900.0)/500.0)**2))*H3
      IF (KN.GT.0) FE = HE

C... Determination of x_max as in Equation (32) of ref [2].
      KM = 10.0*(1.0 - EXPF(-((AI-39.0)/54.5)**2))
      IF (KM.LT.2) KM = 2

      IF (KN.GT.KM) GO TO 100

C... Calculation of cross section as in Equation (31) of ref [2].
      QH = 17.0
      IF (KN.GT.0 .AND. IZ.LE.50) QH = 27
      IF (KN.GT.0 .AND. IZ.GT.50) QH = 21
      IF (KN.EQ.0 .AND. IZ.LE.28) QH = 33
      IF (KN.EQ.0 .AND. IZ.LE.23) QH = 28
      IF (KN.EQ.0 .AND. IZ.LE.5)  QH = 21
      QJ = QH

C... Determination of d as described below Equation (23) of ref [2].
      IF (KN.EQ.0) DX = -3.0
      IF (KN.EQ.1) DX = -1.0

      GO TO 6

C... Calculation of H*(E) as below Equation (34a) in ref [2].
C...
 3    FN = AMIN1((EI/4200.)**(0.72 - 0.18*XN)*H3,1.0)
      IF (IA.LE.70) FE = 1.0 - EXPF(-(EI/35.0)**4)
      IF (IA.GT.70 .AND. KN.LE.4) FE = FN
      IF (IA.GT.70 .AND. KN.GT.4) FE = H3

C... Calculation of H(E) as in Equation (34a) of ref [2].
      IF (EI.LT.200.0) FE = FE*EI/200.0

C... Calculation of x_max as in Equation (35) of ref [2].
      KM = AI/25.0 + 0.5
      XM = KM

      IF (KN.GT.KM) KFLAG = 1
      IF (IA.LE.70.0) KFLAG = 0

C... Calculation of H(E) as in Equation (34a) of ref [2].
      FM = FE
      IF (KN.LE.4) FM = (EI/420.0)**(0.72-0.18*XM)*H3
      IF (FM.GT.1.0) FM = 1.0
      IF (EI.LT.200.0) FM = FM*EI/200.0

C... Calculation of cross section as in Equation (33b) of ref [2].
      QH = 0.2 + 60.0*(EXPF(-((AI - 89.0)/25.0)**2 - ((XN-4.6)/2.0)**2)
     &       + (1.0 - EXPF(-(AI/135.0)**3))*
     &       EXPF(-((XN - AI**0.46)/AI**0.27)**2))
      HM = 0.2 + 60.0*(EXPF(-((AI-89.0)/25.0)**2-((XM-4.6)/2.0)**2)
```

```
     &       + (1.0 - EXPF(-(AI/135.0)**3))*
     &       EXPF(-((XM - AI**0.46)/AI**0.27)**2))

C... Determination of d as described below Equation (23) of ref [2].
      IF (KN.EQ.0) DX = -10.0
      IF (KN.EQ.1) DX = -3.0

      IF (IA.LE.70) GO TO 100
      IF (KN.LE.2) QH = (1.2E5/AJ**2.2)*0.12*EXPF(XN/0.85)
      IF (KN.GE.3 .AND. IZ.LE.26) QH = (1.2E5/AJ**2.2)*EXPF((XN-2)/2.0)
 4    IF ((JZ.LT.82 .AND. KN.GT.KM) .OR. (IZ.GE.88 .AND. JP.GT.3))
     &                               KFLAG = 1
      QH = 0.1**(JP - 3)*QH
      HM = 0.1**(JP - 3)*HM
 6    IF (EI.GE.EC) FE = 1.0

C... Calculation of Y(A,Z) as in Equation (22) of ref [2] and Equation (3)
C... of ref [3].
      IF ((DX*(AI-AM)).LE.0) YA = EXPF(DX*(AI - AM)/ZI)
      IF ((DX*(AI-AM)).GT.0) YA = 2.0 - EXPF(-DX*(AI-AM)/ZI)
      IF (JP.GE.3 .AND. IA.LE.70) YA = 1.0

C... Y(A,Z) includes correction factor phi(A,E) as in Equation (38)
C... of ref [2].
C...
 10   IF (IA.GT.157 .AND. EI.GT.500 .AND. (JP*KN).NE.1)
     &       YA = YA*(1.0 - 0.012*(AI-157.0)*(1.0 -
     &       EXPF(-((EI-500.0)/290.0)**2)))
     &       *(1.0 - EXPF(-0.6*(1.0*JP)**0.8-((92.0-ZI)/2.7)**2))

C... A product with a nucleus at least as large as carbon's or if it's at
C... least as large as beryllium's with an incoming nucleus greater than 88,
C... we'll handle it later.
      IF (JP.GT.5 .OR. (JP.GT.3 .AND. IZ.GE.88)) GO TO 100
      QH = QH*YA

C... Calculation of cross section as in Equation (37) of ref [2].
      IF (IZ.EQ.92 .AND. JZ.EQ.88) QH = 1.2*EXPF(-0.70*ABS(JA - 224.0))
      IF (IZ.EQ.92 .AND. JZ.EQ.89) QH = 1.6*EXPF(-0.15*ABS(JA - 224.0))
      IF (IZ.EQ.92 .AND. JZ.EQ.90) QH = 8.0*EXPF(-0.25*ABS(JA - 233.0))
      IF (IZ.EQ.91 .AND. JZ.EQ.91) QH = 18.5*EXPF(-0.55*ABS(JA - 234.0))
      IF (IZ.EQ.92 .AND. JZ.EQ.9) QH = 55.0*EXPF(-0.80*ABS(JA - 237.0))

      QJ = QH*FE
      HM = HM*YA
      QM = HM*FM
      IF (KFLAG.EQ.1) GO TO 100
      GO TO 200

C... Calculation of E0 as in Equation (3) of ref [1].
 100  EC = E0*AI**E1
      IF (EC.GT.4000.0) EC = 4000.0

C...
C... (see end of subroutine)
      IF (IA.GT.100)  FP = 1. - 0.20*(IA-100)/100.
      IF (IA.GT.180)  FP = 0.84

      ANZX = AMIN1(ANZJ,ANZI)
      ANZC = ANZJ - CN
```

```
      IF (ANZC.GT.1.5) ANZC = 1.5
      AZ = ANZJ
      EE = 450.0/EI

C... Calculation of f_FU(E) as in Equation (20a) of ref [2].
      IF (JA.GT.36 .AND. JZ.LT.88 .AND. IZ.EQ.92)
     &      FF = EXPF(33.0*((ANZJ - 1.36) + 0.00006*EI-1200.0/
     &           (EI+80.0)**2))
      IF (JA.GT.36 .AND. JZ.LT.88 .AND. IZ.EQ.90)
     &      FF = EXPF(16.0*EE**0.6*((ANZJ - CN) + 0.00006*EI -
     &      1200.0/(EI + 80)**2))
      IF (JA.GE.36 .AND. JZ.LT.IZ .AND. IZ.GE.84 .AND. IZ.LT.90)
     &      FF = EXPF(19.0*EE**0.6*((ANZX - CN) - ALOG(5.5/EI**0.07)))

C... Determination of g as on p. 349 of ref [2].
      GE = EE
      IF (EI.LE.200.0) GE = 2.25

C... Determination of h as on p. 349 of ref [2].
      H1 = AJ - 105.0 + 0.68*(207.0 - AI)
      IF (EI.GT.450.0 .AND. EI.LT.600.0) H1 = 0.0006*H1*(600.0 - EI)
      IF (EI.LE.450.0) H1 = 0.0900*H1
      IF (EI.GE.600.0 .OR. AI.LT.180.0 .OR. H1.LT.0) H1 = 0.0

C... Determination of portion of f_F(E) as shown in Equation (13) of ref [3].
      AE = 0.0065*(207 - IA)*EXPF(-ABS(EI - 700.0)/700.0)

C... Calculation of f_F(E) in Equation (5) of ref [2].
      IF (IA.GE.110 .AND. IZ.LE.83 .AND. JZ.LT.IZ)
     &    FF = EXPF(19.0*EE**0.6*((ANZX - CN) - ALOG(5.5/EI**0.07) - AE)
     &       -(((AJ/AI - 0.46)/0.15)**2*GE+H1))
      IF (IA.GE.110 .AND. IZ.LE.83 .AND. JZ.LT.IZ .AND.
     &    (ANZX - CN).LT.1.3) FF =
     &      FF*EXPF((1.3-ANZX + CN)*(350.0/EI)**4/(1.0 + (130.0/EI)**4))
      IF (FF.GT.FMAX) FF = FMAX

C... Determination of f_B(E) as in Equation (14) of ref [2].
      IF (JZ.GE.88) GO TO 110
      EE = EI/EC
      A4 = (AI/63.0)**4
      IF (EE.GE.0.3) FE = EE**(2.2 - 0.01*A4)
      IF (EE.LT.0.3) FE = EE**(2.2 + 0.01*A4)*0.3**(-0.02*A4)
      IF (JZ.GE.5 .AND. JZ.LE.8) FE = EE**1.8
      IF (JZ.GE.9 .AND. JZ.LE.10) FE = EE**2.0
      GF = 1.0
      EX = EI
      IF (EX .LT. 500.0) EX = 500.0

C... Determination of G as in Equation (15) of ref [2].
      IF(JA.GE.18 .AND. JA.LE.30)
     &      GF = EXPF(2600.0*((ANZX - CN) + 0.01*AI - 3.1)/EX)
      IF(JA.GE.31 .AND. JA.LE.56)
     &      GF = EXPF(5600.0*((ANZX - CN) + 0.0035*AI - 1.72)/EX)
      IF(GF.LT.1.0) GF = 1.0

C... f_B(E) as in Equation (14) of ref [2].
      FE = FE*GF
      IF(JZ.LT.30 .OR. IZ.LE.83) GO TO 140
C
C     URANIUM GROUP, ZP .GT. CU
```

```
C
C... As the various references suggest that data for very large nuclei is
C... sparse at best, I have no idea where these next few lines came from,
C... save that they are used to modify the parameter eta (called CJ).
 110  FU = 1.0
      IF (IZ.GE.88 .AND. JZ.GE.82 .AND. (KN.GT.KM .OR. JP.GT.3))
     &      FU = (1.0 - EXPF(-0.6*(1.0*JP)**0.8 - ((92.0 - ZI)/2.7)**2))
     &      *(1.0 - 0.8*(EXPF(-((ZI - ZJ)/5.4)**8)))
      IF (IZ.GE.88 .AND. JZ.GE.57 .AND. JZ.LT.82) FU =
     &     1.0 - 0.36*EXPF(-((82.0 - ZJ)/27.0)**2 - ((92.0-ZI)/2.7)**2)
      CJ = CJ*FU

C... Calculation of cross section as per Equation (18) of ref [2].
 120  IF(JZ.GE.30 .AND. JZ.LE.35)
     &      QH =  6.5*EXPF(-80.0*(AN/ZJ - 1.26)**2)*PN
      IF(JZ.GE.36 .AND. JZ.LE.43)
     &      QH = 10.0*EXPF(-55.0*(AN/ZJ - 1.33)**2)*PN
      IF(JZ.GE.44 .AND. JZ.LE.50)
     &      QH = 10.0*EXPF(-70.0*(AN/ZJ - 1.36)**2)*PN
      IF (JZ.LE.50) GO TO 130

C... This is mostly described in Equation (18d) of ref. [2] including the
C... discussion following the equation about extrapolation into regions of
C... nuclei larger than Z > 55.
      EE = EI - 100
      IF (EE.LT.150.0) EE = 150.0
      HE = 0.03*(ALOG(EE))**2                 ! Equation (20b) of ref [2].
      IF (HE.GT.1.0) HE = 1.0
      ZZ = JZ
      IF (JZ.LT.55) ZZ = 55.0
      HS = (1.5*EXPF(-1.0/ZJ*(ZZ - 55.0)**2) + 3.0)*FU*HE
      HF = 5.0*EXPF(-10.0/ZJ*(ZZ - 55.0)**1.5)
      CON = 80.0 + 220.0*(1.0 - EXPF(-(ZJ - 55.0)**2/4.0))*(JZ/55)
      IF(JZ.GE.51 .AND. JA.LE.155)
     &      QH = HF*EXPF(-CON*(AN/ZJ - 1.48)**2)*PN
     &      + HS*EXPF(-260.0*(AN/ZJ - 1.29)**2)*PN
      IF(JZ.GE.66 .AND. JZ.LE.87) GO TO 140
      IF (IZ.LT.88) GO TO 140
      QJ = QH

C... Correction for Z = 89 and Z = 90; see p. 353 of ref [2].
      IF (IZ.LE.90) QH = 0.6*QH

 130  QF = QH*FF
      QJ = QF
      GO TO 200

C... Calculation of f(A) as in Equations (10) and (13) of ref. [2] (see also
C... discussion in section (f) of ref [3]).
 140  IF (IZ.GE.29 .AND. JZ.EQ.5) FA = EXPF(0.02*(AI - 56)*(ANZC - 0.6))
      IF (IZ.GE.29 .AND. JZ.GE.6 .AND. JZ.LE.8)
     &      FA = EXPF (0.020*(AI - 56.0)*(ANZC-0.7))
      IF (IZ.GE.29 .AND. JZ.GE.9)
     &      FA = EXPF(0.020*(AI - 56.0)*(ANZC-0.9))
      IF (JZ.LE.11) FA = FA*EXPF(-2.5*(ANZC - 1.0))
      IF (FA.LT.1.0) FA = 1.0

C... See Table 1 of ref [2], which determines regions of applicability for
C... the various ways of calculating the cross section.
      IF (IA.GE.110) FM = FF
```

```
      IF (IA.LT.110) FM = FE
      IF(IA.GE.110 .AND. JA.LT.57 .AND. AJ.GT.(0.23*AI))
     &                                 FM = AMAX1(FE,FF)
      IF(IA.GE.110 .AND. JA.LT.57 .AND. AJ.LE.(0.23*AI)) FM = FE

C... Determination of Delta A_c as in Equation (2) of ref [1].
      DMAX = 31.5 + 0.045*(AI - 36.0)*(ALOG(AI) + 1.23)
      AC = DMAX
      AE = AC
      AH = DA
      IF (AH.GT.DMAX) AH = DMAX
      IF (EI.LT.EC) DMAX = 31.5 + D1*(AI - 36)*(ALOG(EI) - D2)
      AE = DMAX
      IF (DA.GT.DMAX) DA = DMAX

C... Calculation of f1 as in Equation (4) of ref [1].
 150  F1 = EXPF(-0.25 + 0.0074*AI)
      IF (EI.GT.750.0) F1 = 1.0 + (F1 - 1.0)*((EC - EI)/(EC - 750.0))**2
      IF (EI.GE.EC) F1 = 1.0

C... Calculation of f2 as in Equation (5) of ref [1].
      F2 = EXPF(1.73 - 0.0071*EI)
      IF (F2.LT.1.0) F2 = 1.0

C... Calculation of parameter P in Table 1D of ref [1].
      PE = P2/EI**P3
      PH = P0/AI**P1

C... Calculation of Cp as below Table 1D of ref [1].
      IF (IZ.GE.20 .AND. IZ.LE.30)
     &      PE = PE*(1.0 - 0.32*EXPF(-((EI - 100.0)/100.0)**2))
      IF (IA.GT.100)
     &   PE = PE*(1.0 - 0.000015*(AI - 100.0)*(EC + 150.0)/(EI + 150.0))

      IF (IA.LE.71 .AND. EJ.GE.EC) PE = PH
      IF (IA.GT.72 .AND. EJ.GT.3000.0) PE = PH
      IF (EJ.GE.EC .AND. EJ.GE.3000.0) PE = PH
      PX = 0.0980*(1000.0/EJ)**(0.819*ALOG(AI) - 2.732)
      IF (EJ.GT.1000.0 .AND. EJ.LT.3000.0)  PE = PX
      PA = PE * AI
      HA = PH * AI

C... Determination of parameter R of Table 1D of ref [1].
      RJ = R0*AJ**(-R1)*(1.0 - 0.4*(IZ/88))
      IF (AJ.LT.40.0) RJ = R2*AJ**R3*(1.0 - 0.4*(IZ/88))

C... Calculation of A' as discussed below Equation (2a) of ref [2].
      IF (AA.LT.AC .AND. JZ.EQ.40) DM = -1.1
      IF (AA.LT.AC .AND. JZ.EQ.42) DM = 0.8

C...
      IF (AA.LT.AC .AND. JZ.EQ.53) DM =-0.5

      IF (AA.LT.AC .AND. JZ.EQ.55) DM = 1.7
      IF (AA.LT.AC .AND. JZ.EQ.60) DM = -1.1
      IF (AA.LT.AC .AND. JZ.EQ.61) DM = -1.3
      IF (AA.LT.AC .AND. JZ.EQ.63) DM = 0.9
      IF (AA.LT.AC .AND. JZ.EQ.64) DM = 0.7
      IF (IZ.GT.76 .AND. JZ.GT.62) DM = DM + 1.0
      AJ = AJ + DM
```

```
C... Determination of parameter S in Table 1D of ref [1].
      S  = S0 - S1*(AI - AM)/ZI

C... This modified version of the atomic number of the secondary nucleus
C... doesn't seem to have a counterpart description in any of the papers.
      ST = ZJ - (S - T2*AJ - T3*AJ*AJ)*AJ
      IF (ST.LT.-1.0) ZA = ABS(ST)**1.30
      IF (ST.GE.-1.0) ZA = ABS(ST)**1.50
      IF (ST.GT. 1.0) ZA = ABS(ST)**1.75

C... Calculation of cross section as in Equation (1) of ref [1].
      Q0 = C1*F1*F2*PE*AI**C2/(1.0 - C3/PA - (C4 - C3/PA)*EXPF(-PA))
      H0 = C1*1.0*PH*AI**C2/(1.0 - C3/HA - (C4 - C3/HA)*EXPF(-HA))
      QR = CJ*PN*EXPF(-PE*DA - RJ*ZA)*Q0
      QH = CJ*PN*EXPF(-PH*AH - RJ*ZA)*H0

C... Evaluation of A0 as discussed in Equation (16) of ref [3].
      AX = 0.5*(S - SQRT(S*S - 4.0*T2*ZJ))/T2 - DM

C... Evaluation of x0 somewhat as in Equation (15) of ref [3].
      KX = (AI - AX) - (JP - 1)
      XP = KX

      IF (IA.GT.70 .AND. KN.GT.KM .AND. KN.LT.KX .AND. KFLAG.EQ.1)
     &                       GO TO 20
      GO TO 195

C... Calculation of f1 as in Equation (4) of ref [1].
 20   F1000 = 1.0 + (EXPF(0.0074*AI - 0.25) - 1.0)*
     &        ((EC-1000.0)/(EC - 750.0))**2

C... Calculation of cross section as in Table 1D and Equation (1) of ref [2].
      Q1000 = CJ*PN*C1*F1000*AI**C2*P1000*EXPF(-P1000*DA-RJ*ZA)
     &        /(1.0 - C3/P1000/AI - (C4 - C3/P1000/AI)*EXPF(-P1000*AI))

C... Cross section calculated with different DELTA A.
      X1000 = Q1000*EXPF(-P1000*(AI - AX))/EXPF(-P1000*DA - RJ*ZA)

C... Yet another correction to the cross section.
      HR = QH*EXPF(-PH*(AI - AX))/EXPF(-PH*AH - RJ*ZA)
      IF (X1000.GT.HR) X1000 = HR

C... Repeat above procedure to calculate different cross section, but for
C... a different energy.
      F500 = EXPF(0.0074*AI - 0.25)
      Q500 = CJ*PN*C1*F500*AI**C2*P500*EXPF(-P500*DA - RJ*ZA)
     &       /(1.0 - C3/P500/AI - (C4 - C3/P500/AI)*EXPF(-P500*AI))
      X500 = Q500*EXPF(-P500*(AI - AX))/EXPF(-P500*DA - RJ*ZA)
      QR = QR*EXPF(-PE*(AI - AX))/EXPF(-PE*DA - RJ*ZA)
      IF (EI.GT.EC) QR = HR
      IF (EI.GE.1000.0 .AND. QR.GT.HR) QR = HR
      IF (EI.LT.1000.0 .AND. EI.GT.500.0)
     &       QR = X500 + (X1000 - X500)*(EI - 500.0)/500.0
      QH = HM + (HR - HM)*(XN - XM)/(XP - XM)
      QJ = QM + (QR - QM)*(XN - XM)/(XP - XM)
      KFLAG = 0
      GO TO 200

195   KFLAG = 0
```

```
      IF (EJ.GT.EC) QR = QH
      IF (EI.GE.3000.0 .AND. QR.GT.QH) QR = QH

C... Repeat procedure described above, but for different circumstances.
      F500 = EXPF(0.0074*AI - 0.25)
      Q500 = CJ*PN*C1*F500*AI**C2*P500*EXPF(-P500*DA - RJ*ZA)
     &      /(1.0 - C3/P500/AI - (C4 - C3/P500/AI)*EXPF(-P500*AI))
      F1000 = 1.0 + (EXPF(0.0074*AI - 0.25) - 1.0)*((EC-1.0E3)
     &      /(EC-750.0))**2
      Q1000 = CJ*PN*C1*F1000*AI**C2*P1000*EXPF(-P1000*DA - RJ*ZA)
     &      /(1.0 - C3/P1000/AI - (C4 - C3/P1000/AI)*EXPF(-P1000*AI))
      F3000 = 1.0 + (EXPF(0.0074*AI - 0.25) - 1.0)*
     &      ((EC - 3.0E3)/(EC -750.0))**2
      Q3000 = CJ*PN*C1*F3000*AI**C2*P3000*EXPF(-P3000*DA - RJ*ZA)
     &      /(1.0 - C3/P3000/AI - (C4 - C3/P3000/AI)*EXPF(-P3000*AI))
      FDAE = AMIN1((AI - AJ)/(0.14*AI)*(EI/1000.0)**(-2.0/3.0),2.0)
      IF (IZ.GE.29 .AND. IZ.LE.83 .AND. EI.LE.1.0E3) QR = QR*FDAE
      IF (IZ.GE.29 .AND. IZ.LE.83 .AND. EI.GT.1.E3 .AND. EI.LE.3.0E3)
     &      QR = 0.5*(3.0*Q1000 - Q3000) + (Q3000 - Q1000)/2000.0*EI
      QJ = QR
      QI = QR

      IF (IZ.LT.88 .OR. JP.GT.3) GO TO 160
      QJ = QR*FU
      GO TO 200

 160  DD = AC - AA

C... Calculation of (N/Z)c as in Equation (3) of ref [2].
      XZ = 1.29 + 0.005*DD + CN

      IF (QH.GT.0.0) RH = QR/QH
      IF (GF.GT.0.0) FR = FE/GF

C... Calculations for different regions as discussed in Table 1 of ref [2].
      IF (IA.GE.69 .AND. RH.LT.FR .AND. EI.LT.300.0) QR = QH*FR
      IF (IZ.GE.76 .AND. IZ.LE.80) FF = FF*AMIN1(1050.0/EI + EI/EC,6.0)
      IF (IA.GE.110 .AND. IA.LE.238) QF = QH*FF*FA
      IF (IA.GE.69 .AND. JA.LE.57) QE = QH*FE*FA
      IF (IA.LT.110. OR. IA.GT.209 .OR. JA.LE.56) GO TO 190
      IF (DD.GE.20) GO TO 170
      IF (IA.LE.125) GO TO 180
      IF (AZ.LE.XZ .OR. JZ.GT.57) GO TO 170
      KZ = 51 + IZ/76 + IZ/80
      IF (AZ.GT.XZ .AND. JZ.LT.KZ) GO TO 180

C... Calculation of exponent gamma as described below Equation (7) of ref [2]
C... and in Equation (19) of ref [3].
      ER = EI**(2.0/3.0)
      IF (ANZC.LT.1.56) GA = (0.03 + (ZJ - KZ)*0.007)*(1.56 - ANZC)*ER
      IF (ANZC.GE.1.56) GA = 0
      IF (GA.GT.1.0) GA = 1.0

C... Calculation of cross section as in Equation (7) of ref [2].
      IF (JZ.GE.KZ .AND. JZ.LE.57 .AND. DD.LT.20.0 .AND. AZ.GT.XZ)
     &      QJ = QR**GA*QF**(1.0 - GA)
      GO TO 200

C... Cross sections for different regions as discussed in Table 1 of ref [2].
 170  QJ = QR
```

```
      GO TO 200

  180 QJ = AMAX1(QR, QF)
      GO TO 200

C... More regions discussed in Table 1 of ref [2].
  190 IF(IA.GE.210 .AND. JA.GE.57) QJ = QF
      IF(IA.GE.110 .AND. JA.LE.56 .AND. AJ.GT.(0.23*AI)) QJ = QH*FA*FM
      IF(IA.GE.110 .AND. JA.LE.56 .AND. AJ.LE.(0.23*AI)) QJ = QE
      IF(IA.LT.110 .AND. IA.GE.69 .AND. DD.GE.0) QJ = QR
      IF(IA.LT.110 .AND. IA.GE.69 .AND. DD.LT.0) QJ = QE
      IF(IA.LT.69) QJ = QR

C... Situation described in Equation (18d) and below in ref [2].
      IF (QJ.EQ.QR .AND. QI.GT.0.0) QH = QH*QR/QI + 0.000001
      IF (IZ.GE.90 .AND. JZ.GE.66)
     &     QJ = HF*EXPF(-CON*(AN/JZ - 1.48)**2)*PN*(0.6 + 0.2*(IZ - 90))
     &      + HS*EXPF(-RJ*ZA)*PN*(0.6 + 0.2*(IZ - 90))

C... Determination of exponent gamma.
      IF (QJ.EQ.QE .OR. QJ.EQ.QF) GA = 0.0
      IF (QJ.EQ.QR) GA = 1.0

      IF (JP.LT.3 .OR. JP.GT.5) GO TO 200

C... Determination of H*(E) as in Equations (34a) and (34b) of ref [2].
      FE = 1.0 - EXPF(-(EI/35.0)**4)
      IF (EI .LT. 200.0) FE = FE*EI/200.0

      IF (JP.GE.3 .AND. KN.GE.1 .AND. IA.LE.70) GO TO 200

C...
      IF (IZ.GT.30.AND.JP.GE.3.AND.KN.GE.KM)  GO TO 200

      QJ = QH*FE*YA

C... Calculation of E' as used in ref [3];
C... justification of equation not found, however.
  200 EX = E0*56.0**E1*EJ/EC

C... Determination of correction factor f1 as in Equation (3) of ref [3].
      G1 = 1.0 - 0.6*(1.0 - EXPF(-(EX/1000.)**2))*EXPF(-(EX/2000.0)**2)
     &     + 0.2*(1.0 - EXPF(-(EX/3000.0)**2))

C... Determination of correction factor f2 as in Equation (9) of ref [3].
      G2 = 1.0 - 0.4*(1.0 - EXPF(-(EX/2000.0)**2))*
     &     EXPF(-((EX - 1800.0)/1800.0)**2)
     &     + 0.17*(1.0 - EXPF(-(EX/2000.0)**2))

C... Determination of correction factor f3 as in Equation (10) of ref [3].
      G3 = 1.0 + 0.25*(1.0 - EXPF(-(EX/1500.0)**2))*
     &     EXPF(-((EX - 1500.0)/1800.0)**2)
     &     - 0.05*(1.0 - EXPF(-(EX/2000.0)**2))

C... Determination of correction factor f4 as in Equation (11) of ref [3].
      G4 = 1.0 - 0.1*(1.0 - EXPF(-(EX/4000.0)**2))

C... Application of correction factors as described in part (b) of ref [3].
      IF (IZ.LE.28 .AND. JZ.LE.4 .AND. JA.LE.12) QJ = QJ*G1
      IF (IZ.LE.28 .AND. AA.GT.AC) QJ = QJ*G2
```

```
      IF (IZ.GT.28 .AND. JA.LE.56 .AND. AA.GT.AC) QJ = QJ*G2
      IF (AA.GE.7.0 .AND. AA.LE.(AC-13.0)) QJ = QJ*G3
      IF (IZ.GE.90 .AND. JA.GT.56 .AND. AA.GE.7.0) QJ = QJ*G4

C... Correction for when the change in the number of neutrons is much larger
C... than the change in the number of protons.
      MX = 10
      IF (IA.LT.150 .AND. JP.LT.3) MX = 9
      IF (IA.GE.150 .AND. JP.EQ.3) MX = 11
      IF (JP.EQ.1 .AND. KN.GT.MX) QJ =
     &                           QJ*(0.1 +0.9*EXPF(-(XN - MX)**2/4.0))
      IF (JP.EQ.2 .AND. KN.GE.MX) QJ =
     &                           QJ*(0.1 +0.9*EXPF(-(XN - MX)**2/4.0))
      IF (JP.EQ.3 .AND. KN.GE.MX) QJ =
     &                           QJ*(0.5 +0.5*EXPF(-(XN - MX)**2/4.0))

C...
      QJ = QJ*FP

      RETURN
      END
C
C
C
C
      SUBROUTINE PXN(Z, A, X, E, QJ)
C.....................................................................
C.....................................................................
C...                                                               ...
C... This subroutine is for the case where the incoming nucleus is ...
C... for elements between scandium and uranium and the secondary   ...
C... nucleus is between boron and the incoming nucleus, i.e.,      ...
C...        21 .LE. IZ .LE. 92   .AND.   5 .LE. JZ .LE. IZ         ...
C...                                                               ...
C.....................................................................
C.....................................................................
      COMMON /A/ Q1, Q2, QI, F1, F2, F3
      EI = 500.0 + 300.0*X
      Q1 = 3.3*EXPF(-ABS(6.9 - X)**2.8/67.45 -
     &          90.0*X**2.35*ABS(2.5 - A/Z)**5)
      QA = 3.3*EXPF(-ABS(6.9 - X)**2.8/67.45 - 0.04638*X**2.35)
      QB = 3.3*EXPF(-ABS(6.9 - X)**2.8/67.45)
      IF (A/Z.GE.2.28 .AND. A/Z.LE.2.50)
     &      Q1 = QA**((2.5 - A/Z)/0.22) * QB**((A/Z - 2.28)/0.22)
      IF (A/Z.GT.2.5) Q1 = QB
      QI = Q1
      XZ = ((A - Z) - X + 1)/Z
      Q2 = 0.5*EXPF(-90.0*ABS(1.5-XZ)**5)
      IF (XZ.LE.1.2) QI = AMIN1(Q1, Q2)
      IF (Z.GE.81.0) FI = 1.5*EXPF(-X*((Z - 80.)/12.0)**5)
      IF (FI.GT.1.0) FI = 1.0
      IF (Z.GE.81.0) QI = QI*FI
      IZ = Z + 0.1
      NX = X + 0.1
      IA = A + 0.1
      NT = IA - IZ
      N  = NT - NX
      ED = E
      IF (NX.GE.3 .AND. IZ.GE.39 .AND. NT.GE.28 .AND. N.LT.50)
     &      ED = E - 10.0
```

```
      IF (NX.GE.3 .AND. IZ.GE.59 .AND. NT.GE.50 .AND. N.LT.82)
     &      ED = E - 3.0
      B = A
      IF (B.LT.35.0) B = 35.0
      FX = (12.0 + 0.1*X)*X - (1.0 - 1.0/X**0.5)*B**(2.0/3.0)
      ALOGY = (1.0 + 1.5*X)*ALOG(ABS(ED/FX))
      IF (ALOGY.GT.10.0) ALOGY=10.0
      F1 = 1.0 - EXPF(-EXP(ALOGY))
      C = 1.0 + 0.03*X*(A-208.0)
      IF (C.LE.1.0) C = 1.0
      D = 27.5 - 0.1*(A + (200.0 - A)/X**0.5)
      IF ((1.0 - X)*(208.0 - A).GT.0.0) D =
     &                                 D - 0.03*(1.0 - X)*(208.0 - A)
      F = E
      IF (NX.GE.3 .AND. IZ.GE.39 .AND. NT.GE.28 .AND. N.LT.50)
     &      F = E - 3.0
      IF (NX.GE.3 .AND. IZ.GE.59 .AND. NT.GE.50 .AND. N.LT.82)
     &      F = E - 3.0
      IF (NX.EQ.1 .AND. IZ.GE.79 .AND. IZ.LE.83)  F = E + 5.0
      F2 = 3500/C * EXPF(-0.6*X - 0.5*(((F - D) - 5.0*X**1.34)/
     &      (6.0 - 2.5/X**4))**2)
     &      * (1.0 - EXPF(-(0.03*A/(2.0 - 1.0/X**4))**3))
      G  =  0.01*(A - 208.0)
      IF (A.LT.208.0) G = 0.0
      F3 = (1300.0/(E + 20.0*X**1.5/E))**(1.3 - G)
      FH = (1300.0/(EI + 20.0*X**1.5/EI))**(1.3 - G)
      FE = F1*(F2 + F3)
      IF (E.GE.EI) QJ = FH*QI
      IF (E.LT.EI) QJ = FE*QI
      IF (Z.EQ.6.0 .AND. (A-X+1.0).EQ.13.0) QJ = QJ*0.4
      IF (Z.EQ.3.0 .AND. (A-X+1.0).EQ.9.0) QJ = QJ*0.65

      RETURN
      END
C
C
C
C
      FUNCTION EXPF(X)
      IF (X.LT.-23.0) X = -23.0
      IF (X.GT.23.0) X = 23.0
      EXPF = EXP(X)
      END
```

```
      SUBROUTINE ZTABLE(ELOWER,EUPPER,M,IZLO,IZUP,TARGET,PSTEP)
C***********************************************************************
C SUBROUTINE ZTABLE in Module UPROP.FOR
C
C Creates the auxiliary ionization loss data file (ZTABLE.DAT).
C
C Modified by AJT 5-8-96 to remove examination of old files.
C
C***********************************************************************
C Parameters
C
C MARR    Maximum number of logarithically-spaced energy bins in spectrum
C ELOWER  Lower energy bound of input and output spectra (>= 0.1 MeV)
C EUPPER  Upper energy bound of input and output spectra (<= 100000 MeV)
C M       Number of logarithmically equally-spaced energy bins (<= MARR)
C IZLO    Least atomic number of elements transported (>= 1)
C IZUP    Greatest atomic number of elements transported (<= 109)
C TARGET  Name of the target shielding material (>= 12 bytes)
C PSTEP   A small pathlength over which 2 nuclear fragmentations are
C         unlikely, typically 0.1 g/cm**2.
C
C Important variables
C
C E       Energy at each grid point after shielding (MeV)
C S       Stopping power at each grid point after shielding (initially)
C         Ratio of stopping powers before and after shielding (modified)
C R       Range at each grid point after shielding (initially)
C         Range at each grid point prior to shielding (modified)
C EP      Energy at each grid point prior to shielding
C SP      Stopping power at each grid point prior to shielding
C RP      Range at each energy EP
C
C Subprograms
C
C SUBROUTINE RANGE(E,M,Z,A,TARGET,R)
C   Returns the range R(M) at M energy grid points E(M) for an element with
C   charge Z and mass A in target material TARGET
C
C FUNCTION STPOW(E,Z,A,TARGET)
C   Returns the stopping power STPOW at energy E for an element with charge
C   Z and mass A in target material TARGET
C
C BLOCK DATA D01
C   Defines the atomic masses of elements in the range 1 <= Z <= 109 and
C   places them in the array AMASS
C
C Data File
C
C ZTABLE.DAT
C   Contains ionization loss data for the transport calculation.
C   Automatically created by this subroutine when needed.
C***********************************************************************
      PARAMETER (MARR=5000)

      REAL*4 E(MARR),S(MARR),R(MARR),EP(MARR),SP(MARR),RP(MARR)
      CHARACTER*12 TARGET,TARGET$
      INTEGER*4 STAT,CREME96_OPEN

      COMMON/MASS/AMASS(109)
```

```
      DATA LMAX,MAXVERS/2,10/
      DATA ELOWER$,EUPPER$,M$,IZLO$,IZUP$,TARGET$,
     &      PSTEP$/0.,0.,0,0,0,' ',0./

C     FORMAT Statements

100   FORMAT(1X,2(1PE10.4,2X),3(I5,2X),A12,2X,1PE10.4)
200   FORMAT((1X,6(1PE10.4,2X)))

c      OPEN(UNIT=11,FILE='USER:ZTABLE.DAT',STATUS='NEW')
      stat = creme96_open('ztable.dat','user',11,'new')

C     Write header

      ELOWER$=ELOWER
      EUPPER$=EUPPER
      M$=M
      IZLO$=IZLO
      IZUP$=IZUP
      TARGET$=TARGET
      PSTEP$=PSTEP
      WRITE (11,100) ELOWER,EUPPER,M,IZLO,IZUP,TARGET,PSTEP
      WRITE (11,'(A)') ' '

C     Compute vector of energies

      DE=(EUPPER/ELOWER)**(1./FLOAT(M-1))
      E(1)=ELOWER
      DO J=2,M-1
        E(J)=E(J-1)*DE
      END DO
      E(M)=EUPPER

C     Compute parameters

      DO J=IZLO,IZUP
        Z=FLOAT(J)
        A=AMASS(J)
        CALL RANGE(E,M,Z,A,TARGET,R)
        DO K=1,M
          S(K)=STPOW(E(K),Z,A,TARGET)
        END DO
        DO K=1,M
          DO KK=K,M
            IF (R(KK).GE.R(K)+PSTEP) GOTO 300
          END DO
          KK=M
300       EP(K)=E(KK)-(R(KK)-R(K)-PSTEP)*S(KK)
          R(K)=R(K)+PSTEP
        END DO

C       Iterate LMAX times to improve estimate of EP

        DO L=1,LMAX
          CALL RANGE(EP,M,Z,A,TARGET,RP)
          DO K=1,M
            SP(K)=STPOW(EP(K),Z,A,TARGET)
            EP(K)=EP(K)-(RP(K)-R(K))*SP(K)
          END DO
        END DO
```

```
C        Compute ratio of stopping powers

         DO K=1,M
           S(K)=SP(K)/S(K)
         END DO

C        Write output to ZTABLE.DAT

         WRITE (11,200) (EP(K),K=1,M)
         WRITE (11,100)
         WRITE (11,200) (S(K),K=1,M)
         WRITE (11,100)

       END DO

C      Close output file and stop

       CLOSE(UNIT=11)
       RETURN
       END
```

```
      SUBROUTINE ZTARGET(LNAME)
      CHARACTER*12 LNAME
c     INCLUDE 'CREME96:ZCOMMON.CMN'
      CHARACTER*12 NAME
      REAL NA(28),IADJ(28),NASPM(28),DENS,ETAD
      INTEGER NZ(28),IGAS,NAS,STAT,CREME96_OPEN
      COMMON /TBLOCK/DENS,ETAD,IGAS,NAS,
     &               NZ,NA,IADJ,NASPM,
     &               NTOTAL,AVGZ,AVGZ2,AVGA,AVGI

!     Check to see if current target is LNAME

      IF (LNAME.EQ.NAME) RETURN

!     Open TARGET.DAT and read data for LNAME

c     OPEN(UNIT=10,FILE='CREME96:TARGET.DAT',
c    &  STATUS='OLD',READONLY,SHARED)
     stat = creme96_open('target.dat','cr96tables',10,'old')
 1    FORMAT(1X,I3)
 2    FORMAT(1X,A12,2X,F9.6,2X,F9.6,2X,I1,2X,I2)
 3    FORMAT(1X,I3,2X,F8.4,2X,F5.1,2X,F9.5)
      READ(10,1) NM
      DO J1=1,NM
        READ(10,2) NAME,DENS,ETAD,IGAS,NAS
        DO J2=1,NAS
          READ(10,3) NZ(J2),NA(J2),IADJ(J2),NASPM(J2)
        END DO
        IF (LNAME.EQ.NAME) THEN
          CLOSE(UNIT=10)
          GO TO 100
        ENDIF
      END DO
      CLOSE(UNIT=10)
      PRINT *,' *****  Target Data not available  *****'
      STOP
 100  CONTINUE

!     Compute material parameters

      NTOTAL=0
      AVGZ=0.
      AVGZ2=0.
      AVGA=0.
      AVGI=0.
      DO J1=1,NAS
        NTOTAL=NTOTAL + NASPM(J1)
        AVGZ=AVGZ + NASPM(J1)*FLOAT(NZ(J1))
        AVGZ2=AVGZ2 + NASPM(J1)*FLOAT(NZ(J1))**2
        AVGA=AVGA + NASPM(J1)*NA(J1)
        AVGI=AVGI + NASPM(J1)*ALOG(IADJ(J1))
      END DO
      AVGZ=AVGZ/FLOAT(NTOTAL)
      AVGZ2=AVGZ2/FLOAT(NTOTAL)
      AVGA=AVGA/FLOAT(NTOTAL)
      AVGI=EXP(AVGI/FLOAT(NTOTAL))
      RAT=AVGZ/AVGA

      RETURN
      END
```

CREME96 ROADMAP

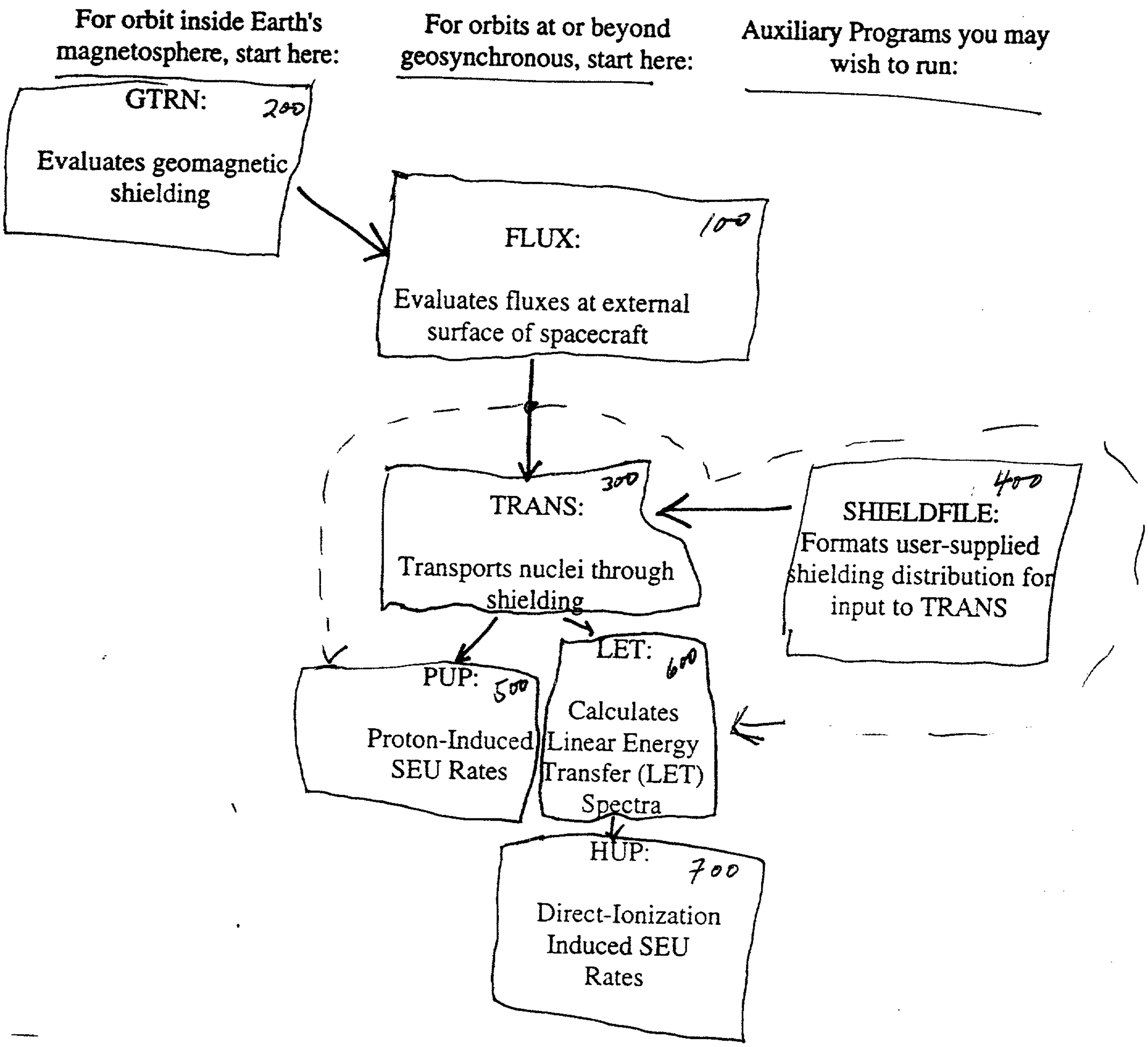


FIG. I

~~Figure 10~~

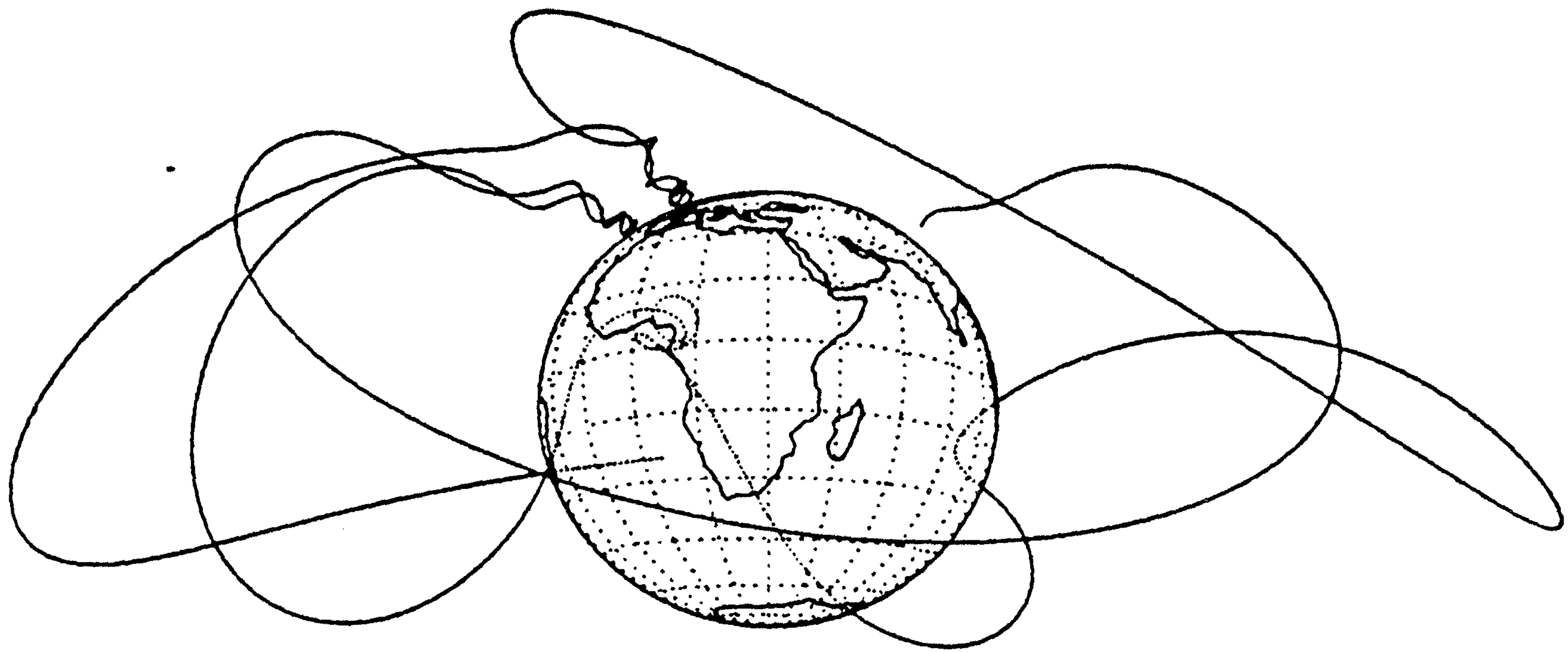


Fig. 5.1 — ~~This~~ figure shows the computed ~~trajectory of a cosmic~~ ray in the earth's magnetic field. ~~Such complex~~ trajectories are not unusual for cosmic ~~rays in~~ the penumbral shadow near the Stormer ~~cutoff~~ (Lund, 1980).

Fig. 2

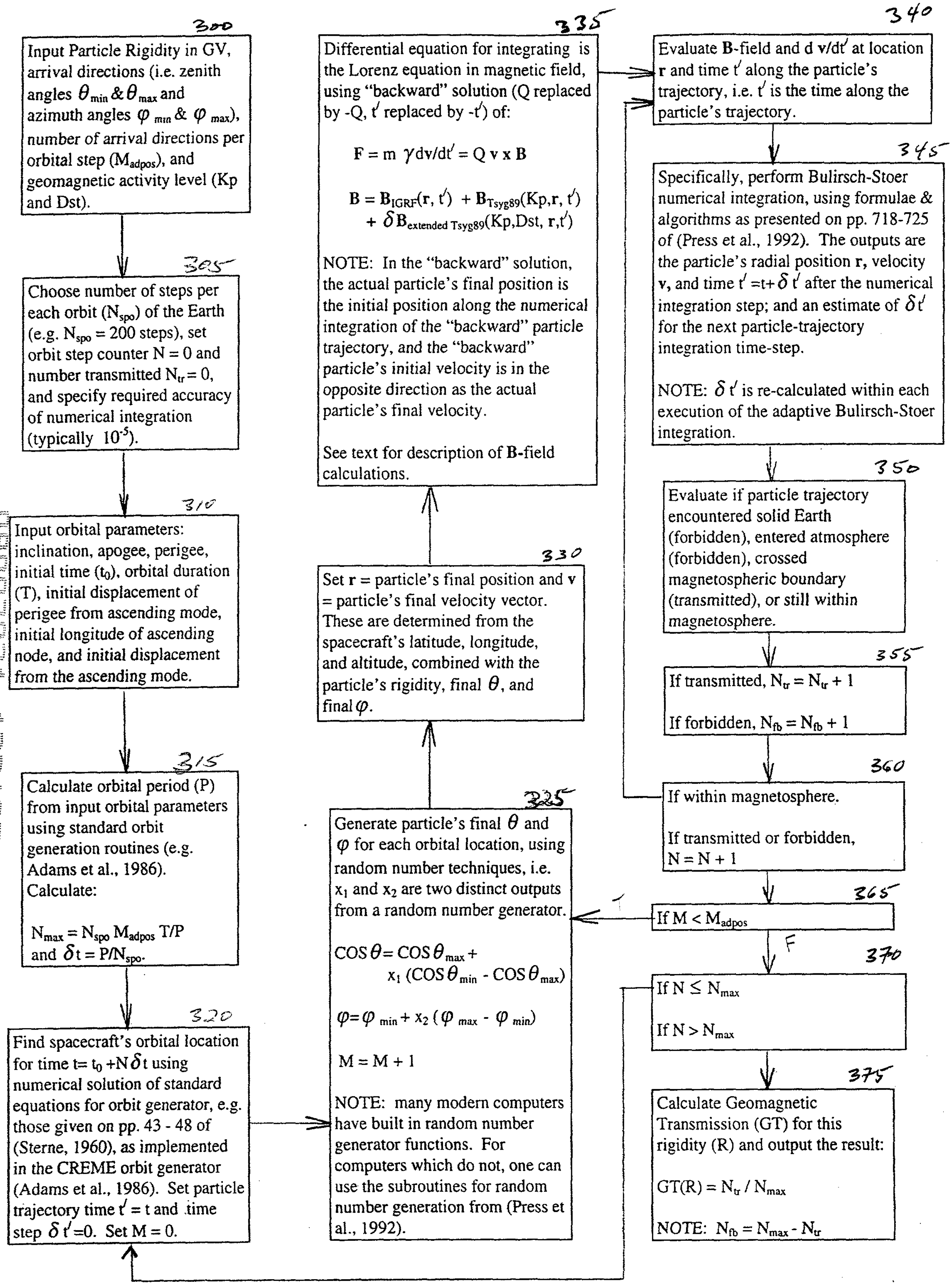
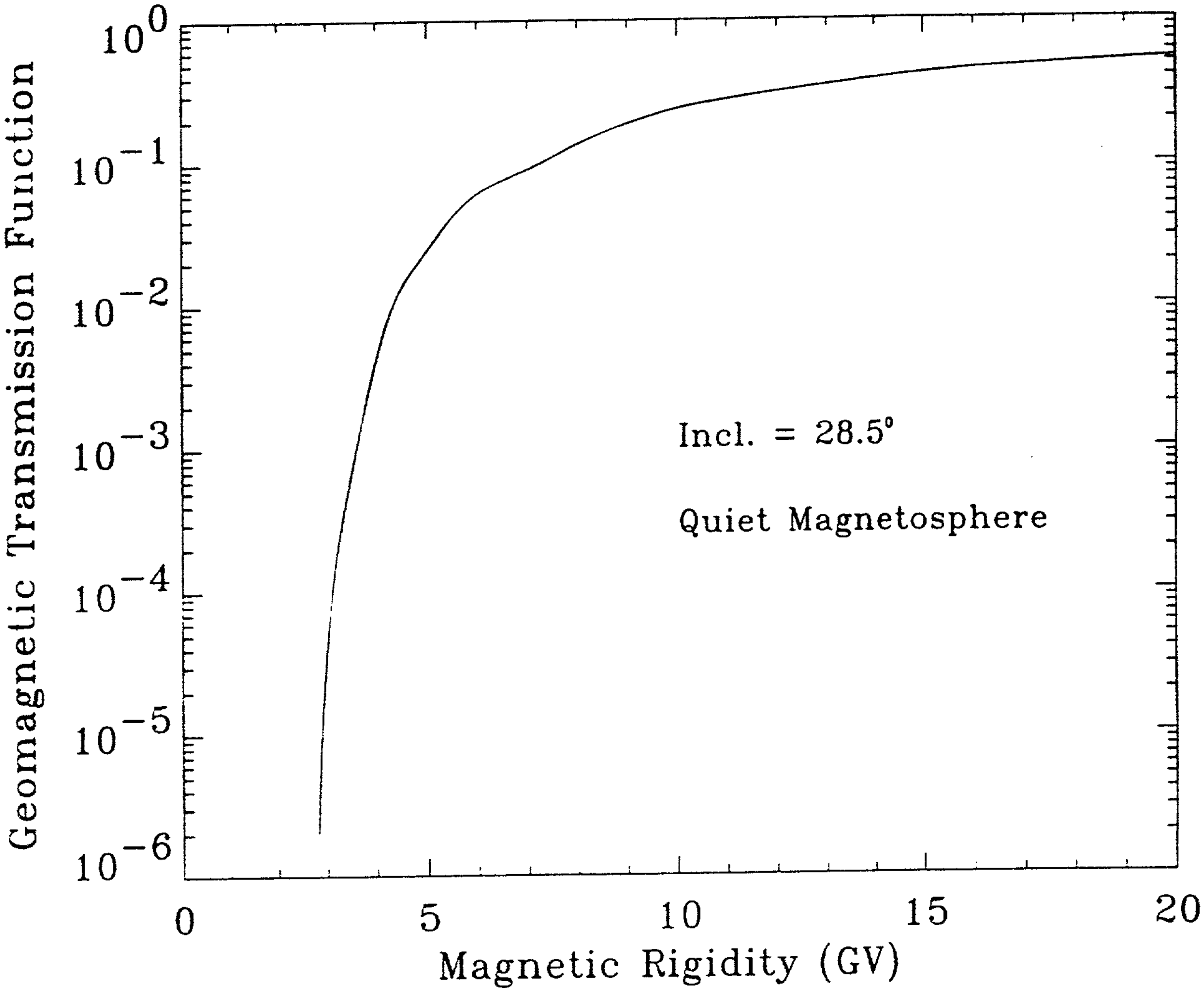

300
Input Particle Rigidity in GV, arrival directions (i.e. zenith angles θ_{min} & θ_{max} and azimuth angles φ_{min} & φ_{max}), number of arrival directions per orbital step (M_{adpos}), and geomagnetic activity level (Kp and Dst).
305
Choose number of steps per each orbit (N_{spo}) of the Earth (e.g. N_{spo} = 200 steps), set orbit step counter N = 0 and number transmitted N_{tr} = 0, and specify required accuracy of numerical integration (typically 10^{-5}).
310
Input orbital parameters: inclination, apogee, perigee, initial time (t_0), orbital duration (T), initial displacement of perigee from ascending mode, initial longitude of ascending node, and initial displacement from the ascending mode.
315
Calculate orbital period (P) from input orbital parameters using standard orbit generation routines (e.g. Adams et al., 1986). Calculate:
$N_{max} = N_{spo} M_{adpos} T/P$ and $\delta t = P/N_{spo}$.
320
Find spacecraft's orbital location for time $t = t_0 + N\delta t$ using numerical solution of standard equations for orbit generator, e.g. those given on pp. 43 - 48 of (Sterne, 1960), as implemented in the CREME orbit generator (Adams et al., 1986). Set particle trajectory time $t' = t$ and time step $\delta t' = 0$. Set M = 0.
325
Generate particle's final θ and φ for each orbital location, using random number techniques, i.e. x_1 and x_2 are two distinct outputs from a random number generator.
$\cos\theta = \cos\theta_{max} + x_1(\cos\theta_{min} - \cos\theta_{max})$
$\varphi = \varphi_{min} + x_2(\varphi_{max} - \varphi_{min})$
M = M + 1
NOTE: many modern computers have built in random number generator functions. For computers which do not, one can use the subroutines for random number generation from (Press et al., 1992).
330
Set **r** = particle's final position and **v** = particle's final velocity vector. These are determined from the spacecraft's latitude, longitude, and altitude, combined with the particle's rigidity, final θ, and final φ.
335
Differential equation for integrating is the Lorenz equation in magnetic field, using "backward" solution (Q replaced by -Q, t′ replaced by -t′) of:
$\mathbf{F} = m\,\gamma\, d\mathbf{v}/dt' = Q\,\mathbf{v} \times \mathbf{B}$
$\mathbf{B} = \mathbf{B}_{IGRF}(\mathbf{r}, t') + \mathbf{B}_{Tsyg89}(Kp, \mathbf{r}, t') + \delta\mathbf{B}_{extended\ Tsyg89}(Kp, Dst, \mathbf{r}, t')$
NOTE: In the "backward" solution, the actual particle's final position is the initial position along the numerical integration of the "backward" particle trajectory, and the "backward" particle's initial velocity is in the opposite direction as the actual particle's final velocity.
See text for description of **B**-field calculations.
340
Evaluate **B**-field and d **v**/dt′ at location **r** and time t′ along the particle's trajectory, i.e. t′ is the time along the particle's trajectory.
345
Specifically, perform Bulirsch-Stoer numerical integration, using formulae & algorithms as presented on pp. 718-725 of (Press et al., 1992). The outputs are the particle's radial position **r**, velocity **v**, and time $t' = t + \delta t'$ after the numerical integration step; and an estimate of $\delta t'$ for the next particle-trajectory integration time-step.
NOTE: $\delta t'$ is re-calculated within each execution of the adaptive Bulirsch-Stoer integration.
350
Evaluate if particle trajectory encountered solid Earth (forbidden), entered atmosphere (forbidden), crossed magnetospheric boundary (transmitted), or still within magnetosphere.
355
If transmitted, $N_{tr} = N_{tr} + 1$
If forbidden, $N_{fb} = N_{fb} + 1$
360
If within magnetosphere.
If transmitted or forbidden, N = N + 1
365
If $M < M_{adpos}$
T
F
370
If $N \le N_{max}$
If $N > N_{max}$
375
Calculate Geomagnetic Transmission (GT) for this rigidity (R) and output the result:
$GT(R) = N_{tr} / N_{max}$
NOTE: $N_{fb} = N_{max} - N_{tr}$

FIG. 3



~~Figure A1~~

FIG. 4

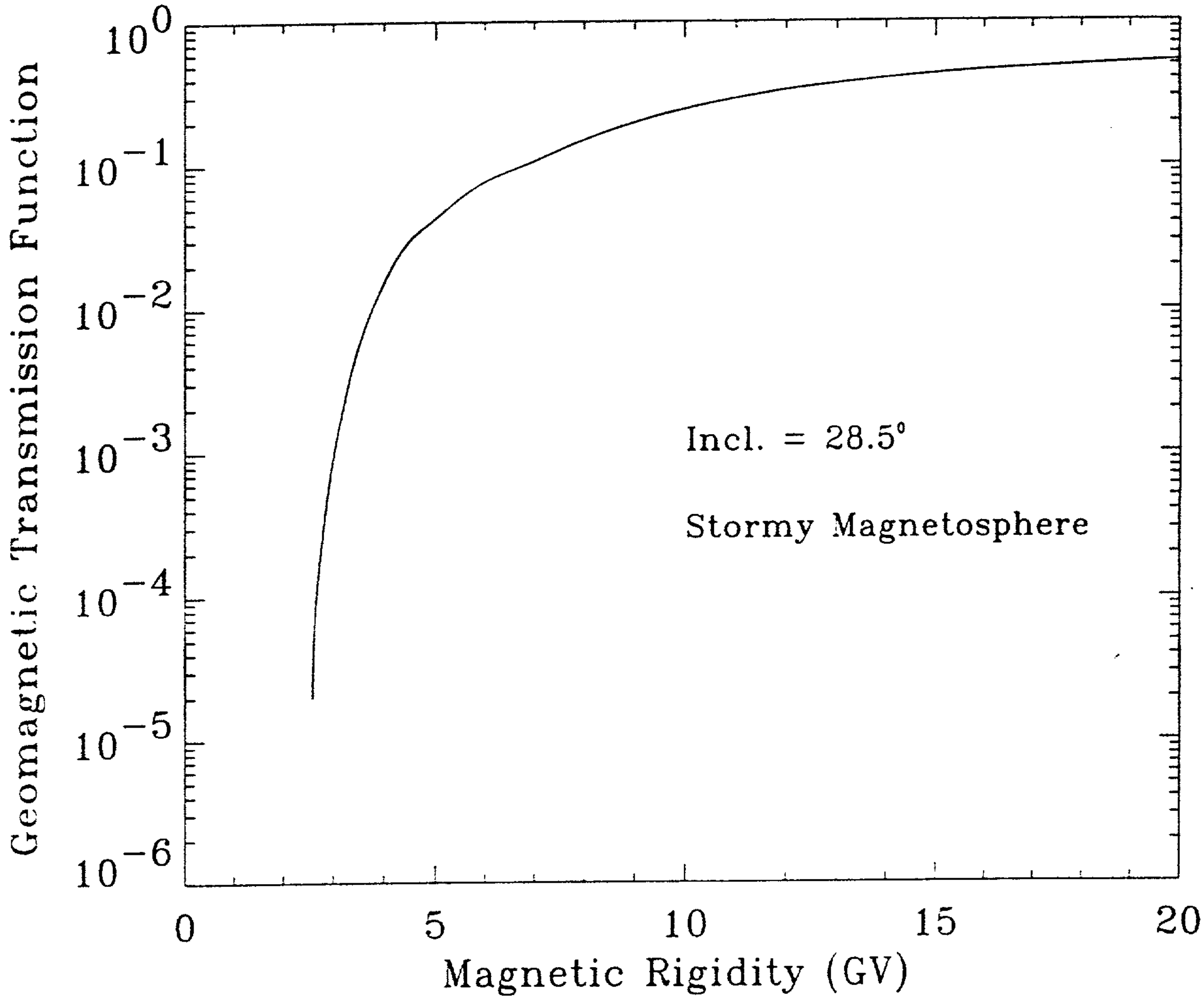


FIG. 5

~~Figure A2~~

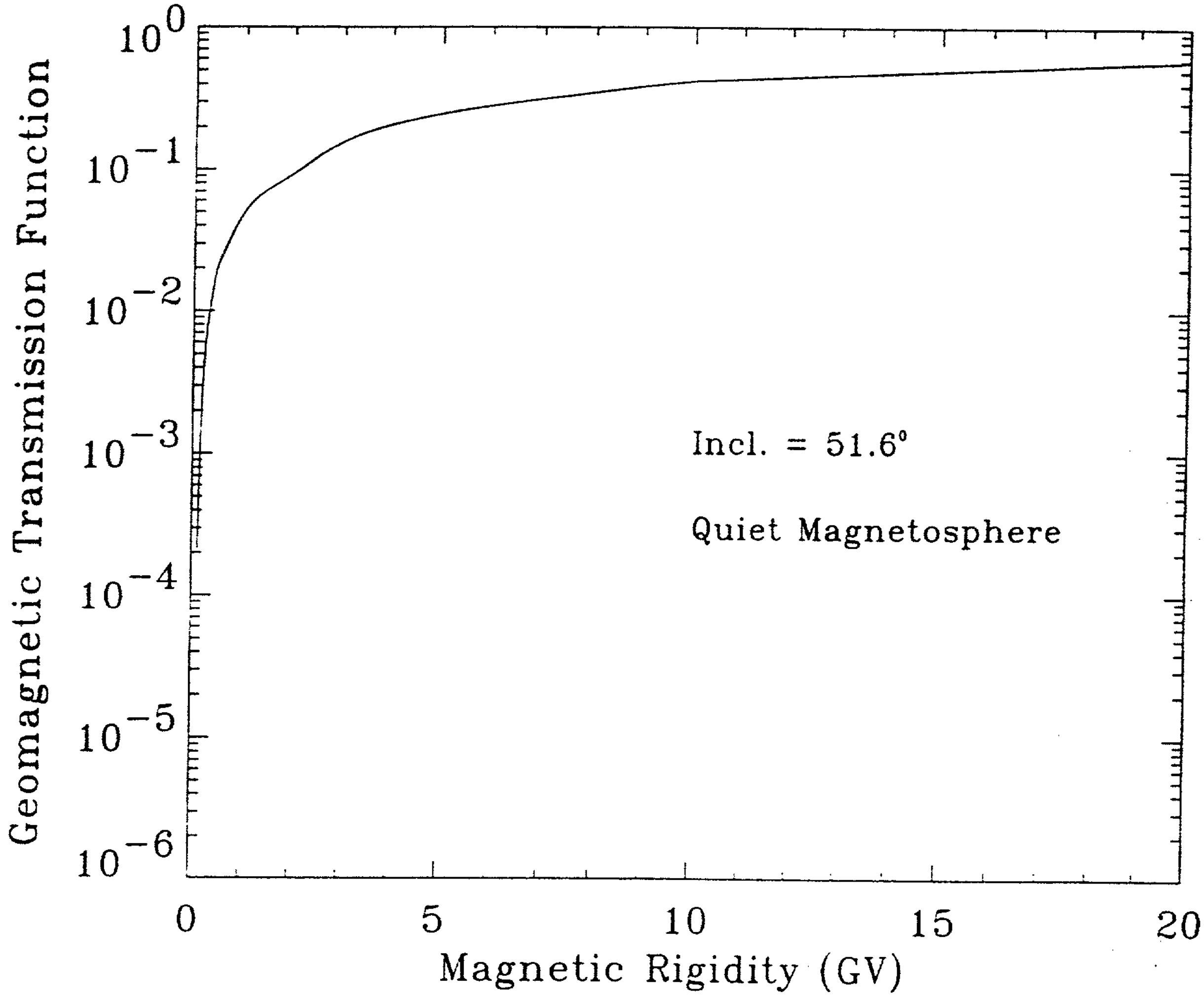


FIG. 6

~~Figure~~ A3

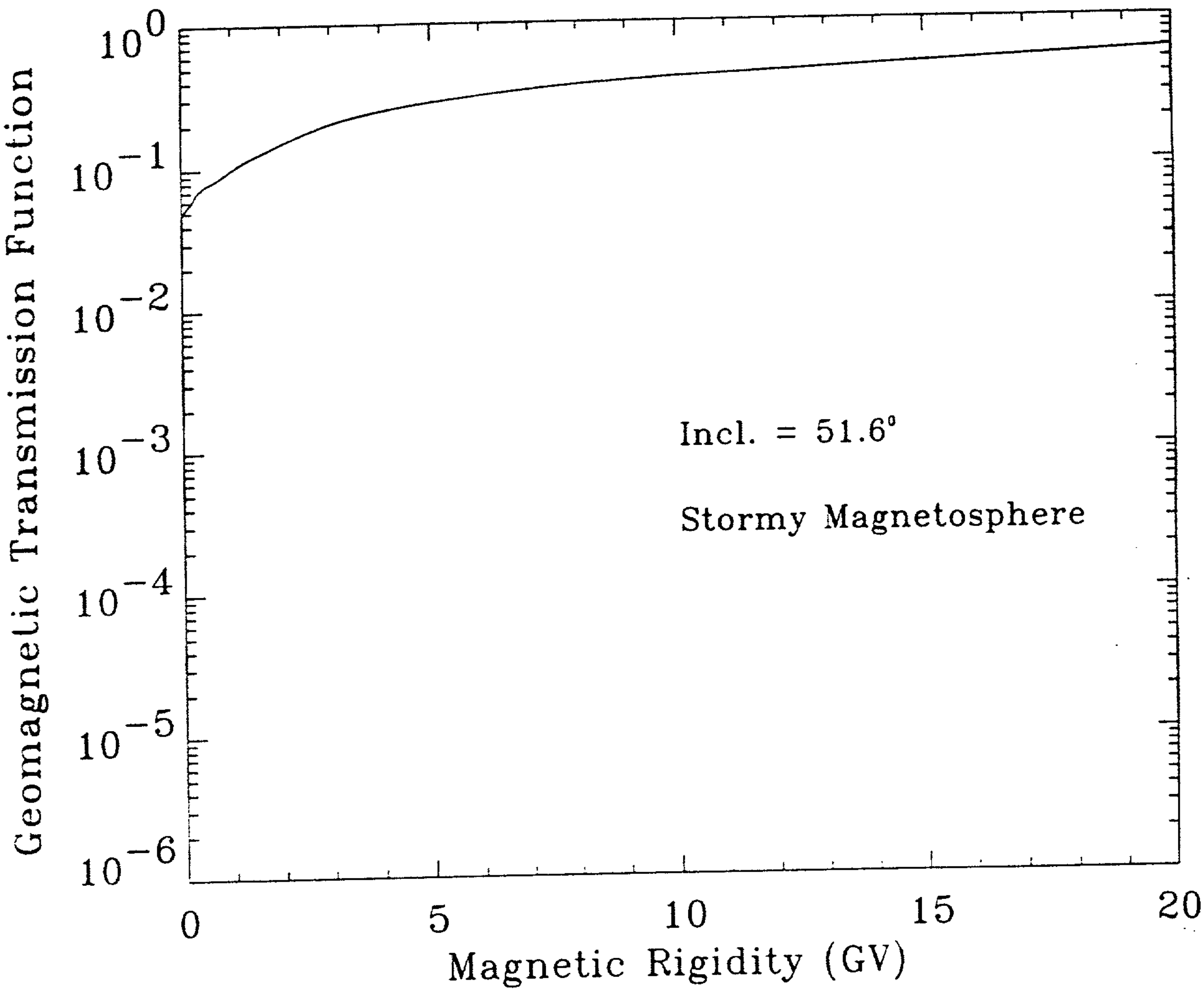


FIG. 7

~~Figure~~ A4

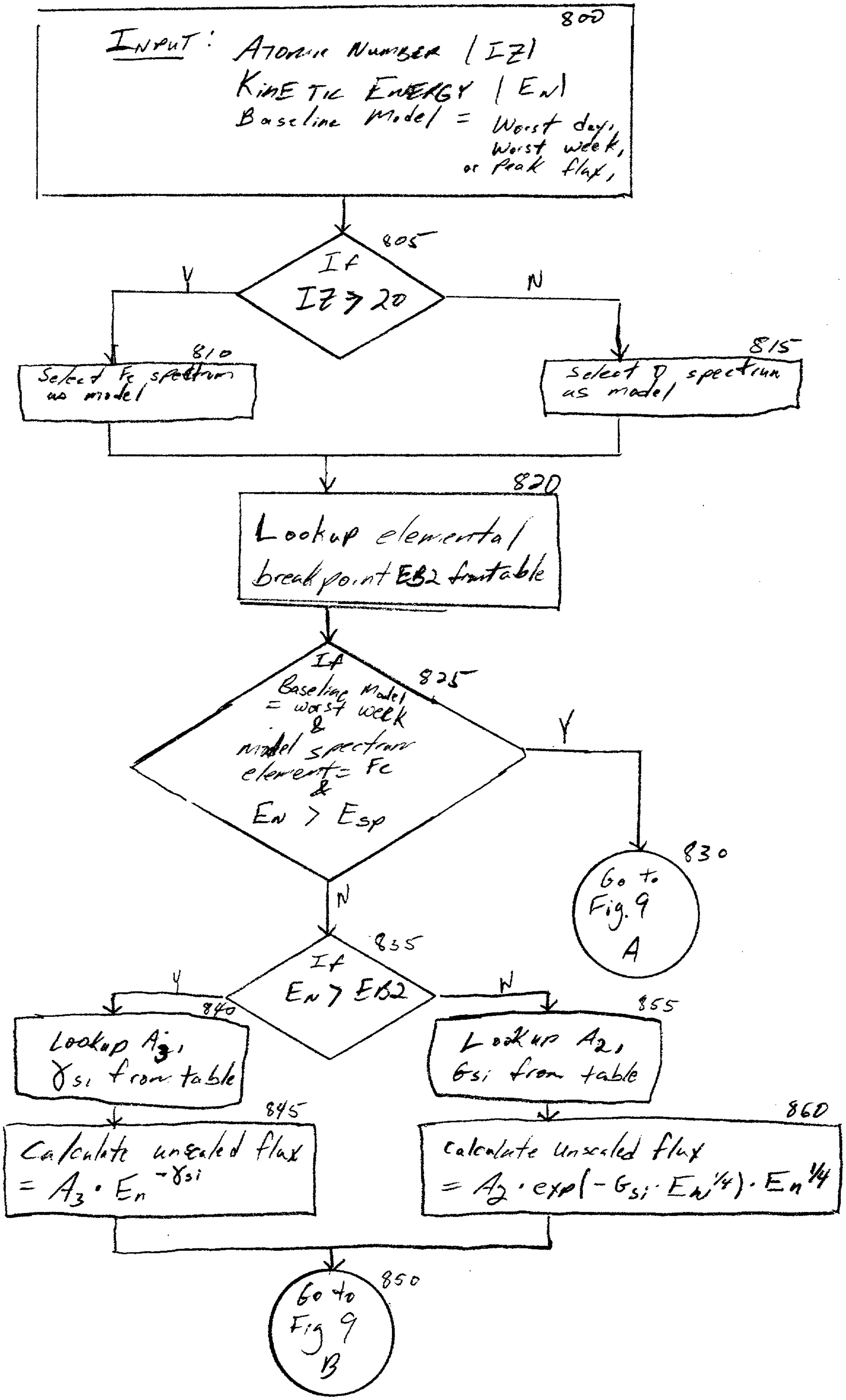

800
INPUT: ATOMIC NUMBER (IZ)
KINETIC ENERGY (EN)
Baseline Model = Worst day, worst week, or peak flux,
805
If IZ > 20
Y
N
810
Select Fe spectrum as model
815
Select D spectrum as model
820
Lookup elemental breakpoint EB2 from table
825
If Baseline Model = worst week & model spectrum element = Fe & En > Esp
Y
830
Go to Fig. 9 A
N
835
If En > EB2
Y
N
840
Lookup A3, γsi from table
855
Lookup A2, Gsi from table
845
Calculate unscaled flux = A3 · En^-γsi
860
Calculate unscaled flux = A2 · exp(-Gsi · En^1/4) · En^1/4
850
Go to Fig 9 B

FIG 8

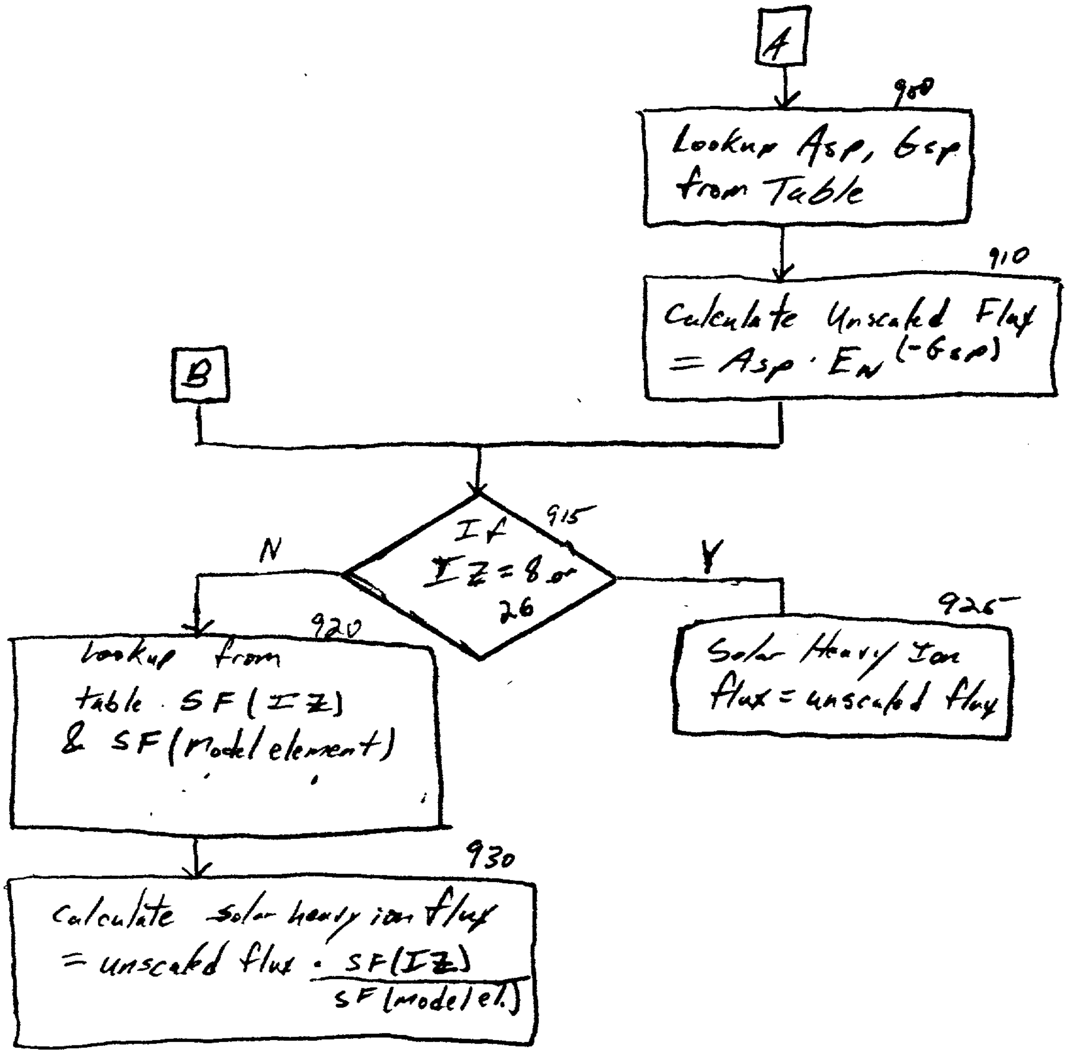

A
900
Lookup Asp, Gsp from Table
910
Calculate Unscaled Flux = Asp · EN^(-Gsp)
B
915
If IZ = 8 or 26
N
Y
920
lookup from table SF(IZ) & SF(model element)
925
Solar Heavy Ion flux = unscaled flux
930
Calculate solar heavy ion flux = unscaled flux · SF(IZ) / SF(model el.)

FIG. 9

CREME96 Main Menu

See brief instructions at the bottom of this page. See How to Run CREME96 for more details.

GO	RESET FORM

Routine	User Request File list	Edit User Request File	Run Routine
TPS		○	☐
GTRN		○	☐
FLUX		○	☐
TRANS		○	☐
LETSPEC		○	☐
PUP		○	☐
HUP		○	☐
UTILTIES (create plots, perform downloads, etc.)			☐

1. To **CREATE** a User Request File:
   - Click on the Edit button by the corresponding routine, leaving the User Request File blank. (If necessary, use the blank space in the pulldown menu to clear a name already typed there.)
   - Click on the "GO" bar.

2. To **EDIT** an existing User Request File:
   - Click on the Edit button by the corresponding routine.
   - Select a User Request File from the appropriate pull-down menu.
   - Click on the "GO" bar.

3. To **RUN** one or more routines:
   - If an Edit button has been clicked "on", use the RESET bar to clear it.
   - Select User Request File(s) from the appropriate pull-down menu(s).
   - Click on the RUN button(s) in that line(s)
   - Click on the "GO" bar.

4. To access **CREME96 Utilities**:
   - Click on the RUN button on that line
   - Click on the "GO" bar.

Note: You can create or edit only one User Request File at a time. But you can submit several files for running (sequentially, in the order shown in the table) by checking more than one "RUN" button and then clicking on the "GO" bar.

Questions? Comments? Send mail to tylka@crs2.nrl.navy.mil.

You may terminate your CREME96 session at any time.

FIG. 10

~~Figure 3~~

FILING DECLARATION
AND POWER OF ATTORNEY

 12/31/97

As a below named inventor, I hereby declare that: My residence, post office address and citizenship are as stated below next to my name. I believe I am the original, first, and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled: METHOD AND APPARATUS FOR MODELING COSMIC RAY EFFECTS ON MICROELECTRONICS, the specification of which is attached hereto.
I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.
I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).
I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign applications for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Number	Country	Filing Date	Priority (Yes/No)

I hereby claim the benefit under Title 35, United States Code, §120 of any United States applications listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of title 35, United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

U.S. Appl. Serial No.	U.S. Filing Date	Status (patented/pending/abandoned)

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys/and/or agent/s/ to prosecute this application and transact all business in the Patent and Trademark Office connected therewith, and hereby certify that the Government of the United States has the irrevocable right to prosecute this application:
Thomas E. McDonnell, Reg. No. 26,950 and John J. Karasek, Reg. No. 36,182.

SEND CORRESPONDENCE TO:
Associate Counsel (Patents), Code 3008.2
Naval Research Laboratory
Washington, D.C. 20375-5000

DIRECT TELEPHONE CALLS TO:
John J. Karasek
Reg. No. 36,182
(202)404-1552

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of joint inventor 1: James H. Adams

Inventor's signature: ______________________ DATE: ____________

Residence:
Citizenship:
Post Office Address:

Full name of joint inventor 2: Paul Boberg

Inventor's signature: Paul R. Boberg DATE: 12-31-97

Residence: College Park, MD
Citizenship: USA
Post Office Address: 5023 Mineola Rd, College Park, MD, 20740

Full name of joint inventor 3: Buddy Brownstein

Inventor's signature: Buddy Brownstein DATE: 12/31/97

Residence: Oxen Hill, MD
Citizenship: USA
Post Office Address: 7402 Oxen Hill Rd, Oxen Hill, MD, 20745

Full name of joint inventor 4: William Dietrich

Inventor's signature: ______________________ DATE: ____________

Residence:
Citizenship:
Post Office Address:

FILING DECLARATION
AND POWER OF ATTORNEY

Navy Case No. 78,824
 12/31/97

Full name of joint inventor 5: Erwin Flueckiger

Inventor's signature: ______________________ DATE: ______________

Residence:
Citizenship:
Post Office Address:

Full name of joint inventor 6: Edward Petersen

Inventor's signature: ______________________ DATE: ______________

Residence:
Citizenship:
Post Office Address:

Full name of joint inventor 7: Margaret Shea

Inventor's signature: ______________________ DATE: ______________

Residence:
Citizenship:
Post Office Address:

Full name of joint inventor 8: Don Smart

Inventor's signature: ______________________ DATE: ______________

Residence:
Citizenship:
Post Office Address:

Full name of joint inventor 9: Edward Smith

Inventor's signature: ______________________ DATE: ______________

Residence:
Citizenship:
Post Office Address:

Full name of joint inventor 10: Allan Tylka

Inventor's signature: ______________________ DATE: ______________

Residence: